SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

INSTITUTO TECNOLÓGICO DE MÉRIDA

TESIS

"DISEÑO DE UN EMULADOR FOTOVOLTAICO BASADO EN APROXIMACIONES DE POLINOMIOS CON SEGMENTACIÓN NO-UNIFORME"

PARA OPTAR AL GRADO DE:

MAESTRO EN INGENIERIA

PRESENTA:

ING. ROSEMBERG OSWALDO RODRÍGUEZ SALAS

ASESOR:

DR. ALEJANDRO ARTURO CASTILLO ATOCHE

MÉRIDA, YUCATÁN, MÉXICO.

15 DE DICIEMBRE DE 2016

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Mérida

DEPENDENCIA: DIV. DE EST. DE POSG. E INV.
No. DE OFICIO: X-571/2016

ASUNTO: AUTORIZACIÓN DE IMPRESIÓN

Merida, Yucatán A 25 de noviembre de 2016

C. ROSEMBERG OSWALDO RODRÍGUEZ SALAS
PASANTE DE MAESTRIA EN INGENIERÍA

De acuerdo al fallo emitido por su asesor el **Dr. José Ramón Atoche Enseñat**, co-dirigido por el **Dr. Alejandro Arturo Castillo Atoche** y la comisión revisora integrada por el Dr. Carlos Alberto Luján Ramírez, y el Dr. Jesús Sandoval Gio, considerando que cubre los requisitos establecidos en el Reglamento de Titulación de los Institutos Tecnológicos le autorizamos la impresión de su trabajo profesional con la TESIS:

"DISEÑO DE UN EMULADOR FOTOVOLTÁICO BASADO EN SEGMENTACIÓN NO UNIFORME DE DOBLE NIVEL Y TÉCNICAS DE ARREGLOS SISTÓLICOS"

ATENTAMENTE
IN HOC SIGNO VINCES

M.C. MIRIAM H. SÁNCHEZ MONROY
JEFA DE LA DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN

S. E. P.
INSTITUTO TECNOLOGICO
DE MERIDA
DIVISION DE ESTUDIOS DE
POSGRADO E INVESTIGACION

C.p. Archivo
MHSM/fjaa.

ABSTRACT: Being similar to Photovoltaic panels, photovoltaic (PV) emulator systems make it possible to perform different PV system tests under various operating conditions. This work outlines a PV emulator design and implementation on a FPGA board. This architecture follows the piecewise polynomial approximation and parallel computing techniques, and shows its capability to generate high-accuracy *I-V, P-V* curves. The main contribution is the addition of hybrid segmentation that enhances speed and area of the implementation maintaining accuracy on the results.

Keywords: piecewise polynomial approximation, segmentation, PV emulator, systolic array network, FPGA.

Agradecimientos

Indice General

1. Introducción 1

1.1 Planteamiento del problema 2

1.2 Objetivo general 3

1.3 Objetivos específicos 3

1.4 Justificación 3

1.5 Alcances y limitaciones 3

1.6 Estado del arte 4

2. Sistemas fotovoltaicos 7

2.1 Introducción 7

2.2 Modelo de un diodo 12

2.3 Modelo de doble-diodo 14

2.4 Mejora al modelo de doble-diodo propuesta por Ishaque 15

3. Modelo fotovoltaico con segmentación no uniforme 19

3.1 Sistemas embebidos 19

3.2 Arreglos sistólicos 21

3.3 Aproximación de polinomios 23

3.3.1 Segmentación no uniforme 27

3.3.2 SQNR 29

3.4 Field Programmable Gate Array (FPGA) 30

4. Hardware de un emulador fotovoltaico basado en aproximación polinomial con segmentación no uniforme 36

4.1 Generación de curva fotovoltaica 37

4.2 Segmentación de doble nivel y aproximación por polinomios 37

4.3 Diseño de evaluador de polinomios utilizando arreglos sistólicos 41

5. Análisis de Resultados 48

5.1 Generación de curva fotovoltaica 48

5.1.1 Casos de estudio 49

5.1.1.1 Celda MSX-60 49

5.1.1.2 Celda Kyocera KG200GT 51

5.1.1.3 SQ150-PC 53

5.1.1.4 Shell SP-70 55

5.1.1.5 Shell ST ST40 57

5.2 Segmentación de doble nivel y aproximación por polinomios 59

5.2.1 Casos de estudio 63

5.2.1.1 Celda MSX-60 63

5.2.1.2 Celda Kyocera KG200GT 66

5.2.1.3 SQ150-PC 68

5.2.1.4 Shell SP-70 70

5.2.1.5 Shell ST ST40 72

5.3 Evaluador de polinomios utilizando arreglos sistólicos 74

6. Conclusiones 78

7. Referencias 80

Tabla de Figuras

Figura 2.1 - Efecto fotovoltaico....8
Figura 2.2 - Componentes de un sistema fotovoltaico....9
Figura 2.3 - Curva I-V Panel FV MSX-60 10
Figura 2.4 (a) Modelo de un diodo para una celda solar ideal (b) modelo de un diodo con Rs....12
Figura 2.5 Curvas características I-V y P-V....13
Figura 2.6 - Modelo de doble diodo para celda fotovoltaica 14
Figura 2.7 - Curva I-V del modelo mejorado y el de doble-diodo para diferentes niveles de irradiación a 25°C 16
Figura 2.8 - Curvas I- V modelo de doble-diodo vs mejorado a diferentes niveles de temperatura a 1KW/m2....17
Figura 3.1 - MPSoC 20
Figura 3.2 - Arreglo sistólico....21
Figura 3.3 Sistema sistólico: (a) Modelo síncrono; (b) Analogía de un arreglo sistólico con el sistema circulatorio 22
Figura 3.4 - Aproximación polinomial. Al incrementar el grado se incrementa la precisión y el número de puntos. 23
Figura 3.5 - Comparación de cuatro tipos de interpolación 25
Figura 3.6 - Segmentación no uniforme aplicada a la raíz inversa 27
Figura 3.7 – Ilustración de Segmentación Uniforme US y tres segmentaciones no uniformes basadas en incrementos o decrementos de potencias de dos: $P2S_L$, $P2_{SR}$ y $P2S_{LR}$ [9] 28
Figura 3.8 Ilustración de una segmentación de doble nivel que es uniforme en ambos niveles, se tienen 8 segmentos de igual longitud externos, y una segmentación interna que aumenta en potencias de dos. Las líneas negras indican la segmentación externa, las líneas grises la interna [9]....29
Figura 3.9 - Estructura de bloques interna del FPGA (www.xillinx.com/fpga/) 31
Figura 3.10 - HDL procesa el código transformandolo en una lista de uniones entre componentes....32

Figura 3.11 - Código en Verilog 33
Figura 3.12 - Los resultados se analizan a través de un Testbench que indica los estados de los componentes en cada instante de tiempo 34
Figura 4.1 - Código en Verilog de la MAC. 42
Figura 4.2 - MAC 43
Figura 4.3 – Pipelines Verilog MAC 44
Figura 4.4 Evaluador de polinomios completo. 46
Figura 4.5 Arquitectura sistema evaluador fotovoltaico 47
Figura 5.1 – Graficas en MATLAB modelación de panel fotovoltaico MSX-60 50
Figura 5.2 - Panel FV Kyocera KG200GT 51
Figura 5.3 - Modulo FV SQ150-PC 53
Figura 5.4 – Shell SP-70 55
Figura 5.5 - Shell ST ST40 58
Figura 5.7 - Coeficientes generados en la aproximación por polinomios segmentación de doble nivel 59
Figura 5.8 - Comparación Punto fijo vs Punto flotante 60
Figura 5.9 - Vista de la aproximación de doble nivel en el intervalo de 0 a 25. [Rojo - Original; Negro - Aproximación punto fijo; Rosa - Aproximación punto flotante] 61
Figura 5.10 - Detalle aproximación de doble nivel. [Rojo – Original ecuación de doble diodo; Negro - Aproximación punto fijo; Azul - Aproximación punto flotante] 61
Figura 5.11 - Error absoluto punto fijo 62
Figura 5.12 - Coeficientes A, B y C en columnas 1, 2 y 3 respectivamente 63
Figura 5.13 - Curva aproximación celda MSX-60 64
Figura 5.14 - Detalle aproximación MSX-60 64
Figura 5.15 - Error absoluto MSX-60 65
Figura 5.16 - Coeficientes A, B y C en columnas 1, 2 y 3 respectivamente 66
Figura 5.17 - Curva aproximación Kyocera KG200GT 66
Figura 5.18 - Detalle aproximación Kyocera 67
Figura 5.19 - Error absoluto Kyocera 67
Figura 5.20 - Coeficientes A, B y C en columnas 1, 2 y 3 respectivamente 68
Figura 5.21 - Curva aproximación SQ150-PC 68
Figura 5.22 - Detalle aproximación SQ150C 69

Figura 5.23 - Error abosluto SQ150C 69
Figura 5.24 - Coeficientes A, B y C en columnas 1, 2 y 3 respectivamente 70
Figura 5.25 - Curva aproximación shell SP-70 70
Figura 5.26 - Detalle aproximación curva Shell SP-70 71
Figura 5.27 - Error absoluto Shell SP-70 71
Figura 5.28 - Coeficientes A, B y C en columnas 1, 2 y 3 respectivamente 72
Figura 5.29 – Aproximación Shell ST40 72
Figura 5.30 - Detalle aproximación Shell ST40 73
Figura 5.31 - Error absoluto Shell ST4 73
Figura 5.32 - Extracto código Testbench 74
Figura 5.33 - Testbench realizado para el modulo MAC 75
Figura 5.34 Análisis del error Testbench MAC 76

1. Introducción

Los sistemas fotovoltaicos se han vuelto una solución popular de generación de energía sustentable debido al bajo mantenimiento que necesitan y la ausencia de partes de desgaste.

La celda solar, principal componente de ésta tecnología se fabrica actualmente con relativa facilidad [1]. Para aprovechar la energía generada se utilizan convertidores de potencia, los cuáles se pueden encontrar en el mercado en diferentes modelos y son una tecnología ya establecida. Sin embargo, cuando se habla de generación de energía fotovoltaica el uso eficiente de la energía solar debe ser asegurado [2].

Actualmente, y sobre todo en nuestro país, se tiene un escaso acceso a la compra o prueba de módulos de energía fotovoltaica, lo cual dificulta el desarrollo de ésta tecnología, ya que para obtener la mayor eficiencia de los módulos se debe contar con una precisa estimación del voltaje y corriente que uno de éstos módulos puede generar bajo ciertas condiciones.

La modelación de módulos fotovoltaicos principalmente, involucra la estimación de curvas *I –V* a través de un modelo matemático, que consiste en una serie de ecuaciones que describen el comportamiento del sistema fotovoltaico; sin embargo, un modelo matemático presenta muchos inconvenientes al no contar con una interfaz directa a etapas de potencia electrónicas quedando únicamente en simulaciones.

Por otra parte, los modelos basados en componentes electrónicos son fáciles de implementar con etapas de potencia, pero incapaces de incorporar la información completa de las condiciones ambientales que los rodean y que afectan de manera importante la precisión de los resultados [3].

Investigaciones previas han utilizado diversas topologías de circuitos para modelar las características principales de los módulos fotovoltaicos cuando éstos son sometidos a variaciones ambientales, como lo son irradiancia y temperatura. La aproximación más simple se conoce como modelo de un diodo [4], [5], que en su versión mejorada incluye un diodo en paralelo a una fuente de corriente y una resistencia en serie [6]; a pesar de la sencillez del modelo, éste exhibe graves deficiencias cuando es sometido a variaciones muy grandes de temperatura.

Para hacer frente a esto y considerar algunas variables físicas involucradas en el modelado de la celda se considera un modelo más preciso conocido como modelo de doble diodo [7]. Sin embargo, la inclusión de un diodo adicional al circuito incrementa la complejidad de la ecuación a modelar a siete parámetros [8].

El principal reto de éste trabajo es estimar los valores de todos los coeficientes manteniendo un bajo esfuerzo computacional, y manteniendo la precisión adecuada para la emulación de éstas curvas en una implementación electrónica.

1.1 Planteamiento del problema

El modelo de doble diodo implica una mejor precisión a la hora de estimar las curvas *I-V*, sin embargo, representa un gran esfuerzo transferir este modelo a un circuito electrónico debido a la complejidad computacional necesaria para calcular los parámetros involucrados.

Utilizando el modelo matemático para doble diodo que existe actualmente, el reto principal consiste en reducir la complejidad computacional involucrada en el proceso utilizando una técnica matemática conocida como aproximación de polinomios, a su vez se pretende implementar el modelo en un hardware específico que contará con arreglos sistólicos y finalmente utilizando la técnica de segmentación no uniforme reducir el número de segmentos computacionales involucrados en la aproximación de la curva *I-V*.

1.2 Objetivo general

Implementar un hardware emulador de sistemas fotovoltaicos que contenga el modelo de doble diodo basado en aproximaciones de polinomios con segmentación no uniforme, en una arquitectura que cuente con arreglos sistólicos, sin sacrificar la precisión del modelado.

1.3 Objetivos específicos

a) Analizar el estado del arte
b) Generar los conocimientos teóricos y prácticos que requiere la implementación del proyecto.
c) Modelar matemáticamente el esquema de doble-diodo.
d) Analizar y proponer una arquitectura reconfigurable en punto fijo.
e) Implementar una arquitectura que utilice segmentación no uniforme.
f) Verificar y validar los resultados arrojados por la implementación propuesta.

1.4 Justificación

Utilizar menos segmentos en la aproximación de las curvas *I*-*V* implicará menos hardware necesario en la implementación del emulador, manteniendo el costo de implementaciones anteriores sin sacrificar el grado de precisión. Esto permitirá en trabajos futuros aumentar la capacidad del emulador para que no solo sea capaz de modelar una celda sino un arreglo de *n* celdas solares.

1.5 Alcances y limitaciones

El presente trabajo estará limitado a la arquitectura de hardware en un FPGA (del inglés Field Programmable Gate Array) que genera los valores correspondientes a las curvas *I* –*V* del sistema. La etapa de potencia necesaria para la implementación completa del sistema emulador no será desarrollada en este trabajo de tesis.

1.6 Estado del arte

Los sistemas fotovoltaicos se han vuelto una alternativa sustentable de generación de energía, éstos sistemas son capaces de generar corriente y voltaje según sus características físicas. Éstos valores de corriente y voltaje varían de acuerdo a ciertos parámetros físicos como irradiancia, temperatura, resistencia entre otros y para describir su comportamiento existen diferentes tipos de modelos matemáticos que los describen.

El modelo de un diodo descrito en [9] explica que la salida de un módulo fotovoltaico es altamente no lineal, por lo cual un elemento no-lineal como el diodo debe ser considerado en el circuito equivalente del diagrama; a su vez, demuestra que existen parámetros de la celda no son constantes y que cambiarán con las condiciones ambientales tales como la irradiancia y la temperatura. Para algunas aplicaciones el modelo de un diodo ofrece una estabilidad adecuada entre simplicidad y precisión, sin embargo, decae ante variaciones de temperatura e irradiancia en los puntos límite de la celda.

Para enfrentar éste problema un modelo más preciso formulado en [10] agrega un par de resistencias extra R_p y R_s y un diodo adicional mejorando sustancialmente las características del modelo contra su predecesor, la complejidad surge al estimar todos los valores de parámetros del modelo mientras se mantiene un tiempo razonable de simulación.

Para enfrentarse a éste problema han existido diversos tipos de aproximaciones, simuladores a través de Simulink en MATLAB, emuladores utilizando microcontroladores low-cost, y emuladores utilizando FPGA.

Como se observa en [8], [11], [12], [13] la simulación de los sistemas fotovoltaicos es el camino utilizado en el caso de que el punto de máxima potencia cambia continuamente, debido a variaciones ambientales. Los criterios buscados por un sistema simulador de sistemas fotovoltaicos son: debe ser preciso al graficar las curvas características de *I-V* y de *P-V* incluyendo sombreado parcial; debe ser una herramienta flexible para desarrollar y validar diseñis de sistemas fotovoltaicos, incluyendo el convertidor de potencia y el control de máximo punto de potencia. Existen diferentes tipos

de software simulador como P-Spice, PV-DesignPro, SolarPro, PVcad y PVsyst en el mercado, sin embargo son relativamente caros, innecesariamente complejos y raramente incluyen la interfaz del convertidor de potencia.

Por su parte los sistemas low-cost basados en microcontroladores como lo son [14] y [15] nos presentan una alternativa de bajo costo, y basándose en la optimización del máximo punto de potencia, sin embargo éstas aproximaciones cuentan con errores mayores al 5% de precisión quedando muy por debajo respecto a otras alternativas.

Los emuladores basados en una arquitectura FPGA como en [16], [17] y [18] presentan ventajas respecto a los anteriores, la facilidad de una conexión a una etapa de potencia, la velocidad y reconfiguración capaz de realizarse por el FPGA además de las diferentes técnicas de implementación que se pueden utilizar para lograr el propósito. En [17] se enumeran algunas técnicas para realizar la implementación de un sistema fotovoltaico en un FPGA vía redes neuronales, alternativa atractiva pero poco eficiente al momento de entrenar y optimizar la red para su funcionamiento.

Modelos basados en CORDIC y LUT fueron analizados, notando buenos resultados en precisión sobre el punto de máxima potencia, sin embargo, en el primero se denota una fuerte carga computacional en procesador para llegar al objetivo al estar basado en descomposiciones iterativas. El modelo de LUT presenta una fuerte carga en espacio de memoria utilizada al tener que almacenar todos los valores posibles para cierto grado de precisión.

En éste trabajo se presentará una propuesta basada en aproximaciones polinomiales que garantizará un grado de precisión alto medido a través de un SQNR significativo y siempre buscando una reducción el área de implementación dentro del FPGA. Esto logrará sentar una base para trabajos futuros en donde al tener un área mínima de implementación se podrá trabajar el punto de máxima potencia de un arreglo completo de $n \times n$ celdas solares utilizando únicamente un FPGA.

2. Sistemas fotovoltaicos

En éste capítulo se describirán los sistemas fotovoltaicos, las magnitudes características de cada uno de ellos, así como dos distintos modelos matemáticos que describen su comportamiento. Se describirán los modelos de un diodo y de doble-diodo, así como se presentarán las curvas *I-V* y las partes que las conforman.

2.1 Introducción

La radiación solar puede ser transformada directamente en energía eléctrica. A este fenómeno se lo denomina efecto fotovoltaico. A mediados del siglo XIX (1839) el físico francés Becquerel descubrió el efecto fotovoltaico (FV). Varios físicos, como Willbughby Smith (1873) y Lenard (1900) verifican su existencia bajo diversas condiciones. Einsten (1905) proporciona la base teórica del fenómeno, ganando el premio Nobel de física. Millikan (1920), un físico norteamericano, corrobora la teoría de Einsten. Sin embargo, la aplicación práctica de esta conversión de energía no comenzó hasta 1954, cuando se necesitó una fuente generadora de energía eléctrica que pudiere alimentar los circuitos eléctricos de los satélites espaciales, sin recurrir al uso de combustibles y con una vida útil de larguísima duración.

La palabra fotovoltaico(a) está formada por la combinación de dos palabras de origen griego: foto, que significa luz, y voltaico que significa eléctrico. El nombre resume la acción de estas células: transfomar, directamente, la energía luminosa en energía eléctrica. El voltaje de una célula FV es de corriente continua (CC). Por lo tanto, hay un lado que es positivo y otro negativo. Para celdas de silicio, este voltaje es de alrededor de 0,5 V.

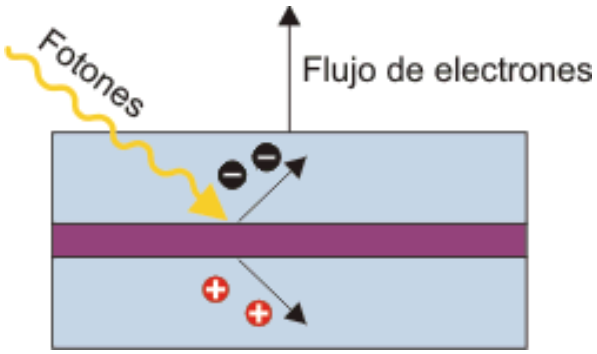


Figura 2.1 - Efecto fotovoltaico

En un instante determinado, la potencia eléctrica proporcionada por la celda FV está dada por el producto de los valores instantáneos del voltaje y la corriente de salida. Este valor es afectado por el comportamiento intrínseco de un material semiconductor, por el nivel de irradiación luminosa, y el método de fabricación de la celda.

Las celdas FV que se ofrecen en el mercado actual utilizan dos tipos de materiales semiconductores. Uno tiene una estructura cristalina uniforme, el otro una estructura policristalina. El tipo cristalino requiere un elaborado proceso de manufactura, que insume enormes cantidades de energía eléctrica, incrementando substancialmente el costo del material semiconductor. La versión policristalina se obtiene fundiendo el material semiconductor, el que es vertido en moldes rectangulares. Los dos tipos pueden ser identificados a simple vista, ya que la estructura cristalina provee una superficie de brillo uniforme, mientras que la policristalina muestra zonas de brillo diferente.

Si los valores de potencia luminosa y la orientación del panel permanecen constantes, la corriente de salida de un panel FV varía con el valor del voltaje en la carga y su temperatura de trabajo. Esto se debe a las características intrínsecas de los semiconductores. La Figura 2.3 muestra, en forma gráfica, la relación entre la corriente y el voltaje de salida para un panel FV (curva *I-V*) para cuatro temperaturas de trabajo, cuando el nivel de radiación permanece constante.

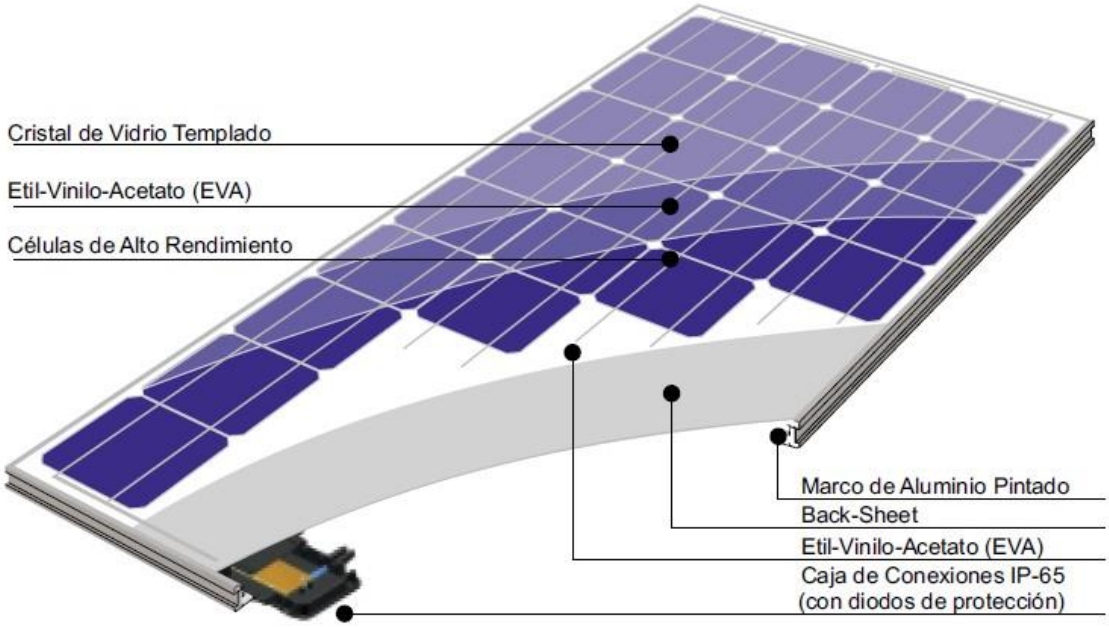


Figura 2.2 - Componentes de un sistema fotovoltaico

Puede observarse que el valor máximo para el voltaje de salida corresponde a un valor de corriente nulo (voltaje a circuito abierto), mientras que el valor máximo para la corriente corresponde a un voltaje de salida nulo (salida en corto circuito). Todas las curvas tienen una zona donde el valor de la corriente permanece prácticamente constante para valores crecientes del voltaje de salida, hasta que alcanzan una zona de transición.

A partir de esta zona, pequeños aumentos en el voltaje de salida ocasionan disminuciones altas en el valor de la corriente de salida. El comienzo de la zona de transición se alcanza para menores valores del voltaje de salida cuando la temperatura de trabajo se incrementa

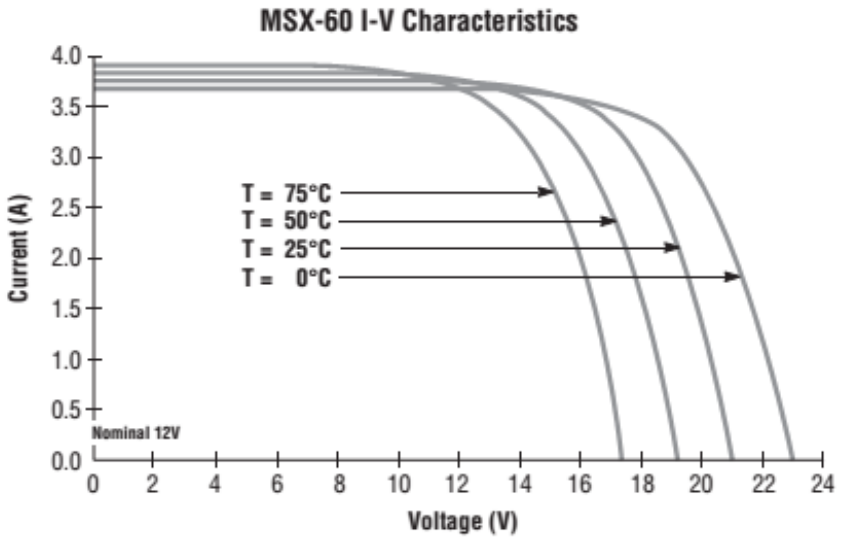


Figura 2.3 - Curva I-V Panel FV MSX-60

La corriente de cortocircuito I_{sc} es uno de los parámetros principales al analizar un panel solar, este parámetro se refiere a la intensidad máxima de corriente que se puede obtener de un panel solar fotovoltaico, el panel debe estar sin ninguna otra resistencia adicional provocando un cortocircuito. Al no existir resistencia al paso de la corriente el voltaje es cero. (Martínez J., 2012, pág. 202)

Otro de los parámetros principales en el funcionamiento de un panel solar fotovoltaico es el voltaje de circuito abierto V_{OC}, el cual es la tensión máxima disponible de una celda solar medida en condiciones de circuito abierto o condiciones de resistencia máxima.

El punto de potencia máxima de un panel solar FV es el parámetro que expresa el punto de funcionamiento para el cual la potencia máxima es entregada.

$$P_{max} = V_{OC} I_{SC} FF \qquad (1)$$

Tomando en cuenta la Ecuación 1 el punto de potencia máxima se obtiene multiplicando el voltaje máximo por la corriente máxima por el factor de forma dando un valor de potencia en Watts indicando que cuando el panel opera en estas condiciones se obtiene el mayor rendimiento posible.

Para cada condición de trabajo se puede calcular la potencia de salida del panel multiplicando los valores correspondientes al voltaje y la corriente para ese punto de la curva *I-V*. En particular, la potencia de salida es nula para dos puntos de trabajo: circuito abierto y cortocircuito, ya que la corriente o el voltaje de salida es nulo. Por lo tanto, si la salida de un panel es cortocircuitada, éste no sufre daño alguno. Entre estos dos valores nulos, la potencia de salida alcanza un valor máximo que varía con la temperatura.

Cuanto la temperatura de trabajo de un panel FV aumenta, el valor de la potencia de salida disminuye, esto debido a que tanto la corriente de corto circuito como el voltaje de circuito abierto, se ven afectados por la misma, pero el tipo de variación, así como su magnitud porcentual, son distintos para estos dos parámetros. Si tomamos referencia la hoja de datos del Panel FV MSX-60 observamos que la corriente de corto circuito aumenta moderadamente (0.065±0.015)%/°C mientras que el voltaje a circuito abierto disminuye considerablemente . –(0.5±0.05)%/°C.

2.2 Modelo de un diodo

El modelo de un diodo para una celda solar ideal (Figura 2.4), consiste en un diodo conectado en paralelo con una fuente de corriente (I_{pv}) cuyo valor depende la luz recibida.

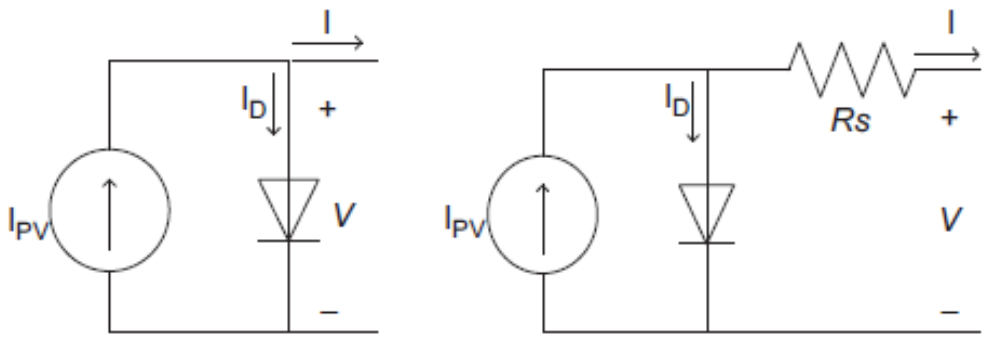


Figura 2.4 (a) Modelo de un diodo para una celda solar ideal (b) modelo de un diodo con Rs

La corriente de salida para Figura 2.4(b) es $I = I_{pv} - I_D$y que se puede escribir como:

$$I = I_{pv} - I_o[\exp\left(\frac{V+IR_s}{aV_T}\right) - 1] \qquad (2)$$

donde I_{pv} es la corriente generada por la incidencia de luz, I_o es la corriente de saturación inversa, V_T es el voltaje térmico de la celda fotovoltaica. La Figura 2.5 muestra las curvas *I-V* y *P-V* características generadas en (2). Generalmente, tres puntos se toman en condiciones de prueba estándar, (I_{sc}, 0), (V_{mp}, I_{mp}) y (V_{oc}, 0) y se pueden encontrar en la hoja de datos del fabricante. Una estimación adecuada de éstos puntos es la principal meta de cualquier técnica de modelado.

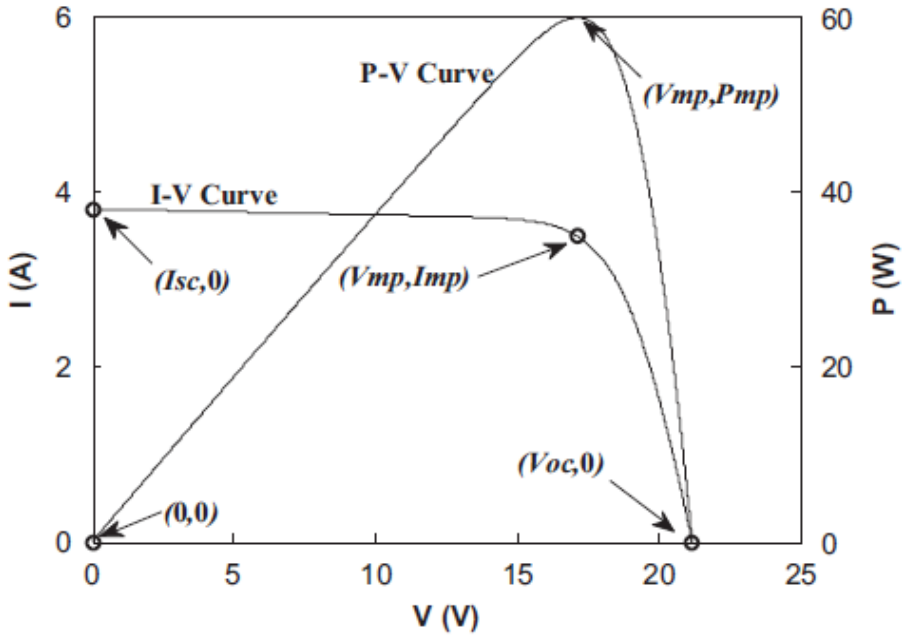


Figura 2.5 Curvas características *I-V* y *P-V*

Sin embargo, (2) no representa adecuadamente el comportamiento de la celda fotovoltaica cuando es sujeta a variaciones ambientales, especialmente en bajos voltajes.

2.3 Modelo de doble-diodo

Un modelo más preciso se muestra en Figura 2.6. La (2) describe la corriente de salida de la celda:

$$I = I_{PV} - I_{d1} - I_{d2} - (\frac{V+IR_s}{R_p}) \quad (3)$$

donde

$$I_{d1} = I_{01}[\exp\left(\frac{V+IR_s}{a_1 V_{T1}}\right) - 1] \quad (4)$$

y

$$I_{d2} = I_{02}[\exp\left(\frac{V+IR_s}{a_2 V_{T2}}\right) - 1] \quad (5)$$

En las ecuaciones (4) y (5) I_{o1} y I_{o2} son las corrientes de saturación inversa del diodo 1 y del diodo 2 respectivamente, V_{T1} and V_{T2} son los voltajes térmicos de los diodos. a_1 y a_2 representan las constantes de diodo ideal. El término I_{o2} en (5), compensa las perdidas por recombinación en la región de agotamiento como se describe en [7].

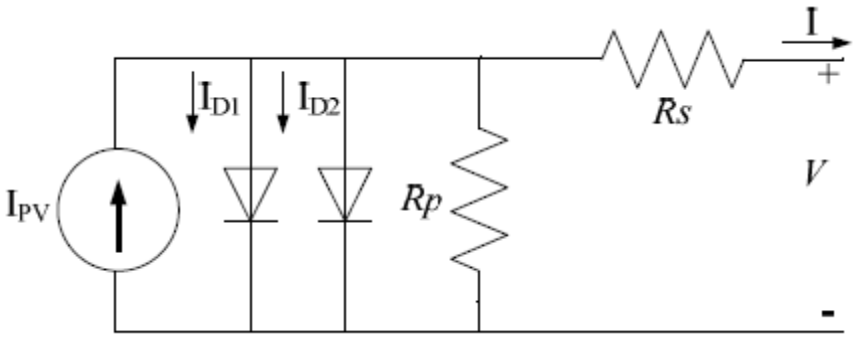


Figura 2.6 - Modelo de doble diodo para celda fotovoltaica

A pesar de que una gran precisión se puede alcanzar usando este modelo, requiere la evaluación de siete parámetros, I_{pv}, I_{o1}, I_{o2}, R_p, R_s, a_1 y a_2.

2.4 Mejora al modelo de doble-diodo propuesta por Ishaque.

En [2] se hace una mejora sustancial al modelo de doble-diodo de 7 parámetros, el principal reto de éste modelo es la complejidad computacional que existe al iterar 7 variables, se busca estimar todos los valores del modelo con un bajo esfuerzo computacional.

La principal contribución de éste modelo es no tratar de resolver la corriente de saturación de los diodos, se establece que ambas corrientes de saturación son iguales en magnitud obteniendo:

$$I_{o1} = I_{o2} = I_o = \frac{(I_{SC_{STC}} + K_I \Delta T)}{\exp[\frac{V_{OC,STC} + K_V \Delta T}{\left\{\frac{a_1 + a_2}{p}\right\} V_T} - 1}$$

Ésta igualdad simplifica la computación debido a que no se requiere iteración; la solución se puede obtener analíticamente. El factor de idealidad de ambos diodos (a_1 y a_2) representan la difusión y recombinación de las componentes de las corrientes respectivamente. De acuerdo a la teoría de difusión de Shockley la corriente de difusión a_1 debe ser la unidad. El valor de a_2 es flexible, basados en simulación extensiva se encontró que un valor mayor o igual a 1.2 es la más aproximada entre el modelo propuesto y la curva *I – V práctica*.

Los últimos dos parámetros (R_p y R_s) son obtenidos a través de iteración. Muchos investigadores han evaluado estos dos parámetros independientemente, pero los resultados no han sido favorables. En el modelo mejorado [2] éstos parámetros se calculan simultáneamente, la idea es aproximar vía el punto de máxima potencia; para llegar al valor calculado de potencia y el experimental obtenido por la hoja de datos se empieza por incrementar iterativamente el valor de R_s mientras simultáneamente se calcula el valor de R_p.

Las condiciones iniciales de ambas resistencias quedan de la siguiente forma:

$$R_{SO} = 0; \qquad R_{PO} = \frac{V_{mp}}{Iscn - Vmp} - \frac{Vocn - Vmp}{Imp}$$

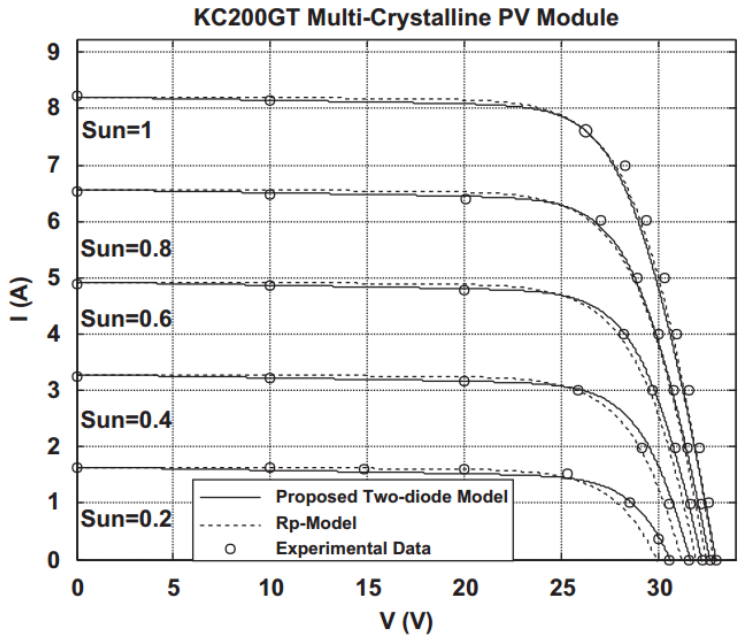


Figura 2.7 - Curva *I-V* del modelo mejorado y el de doble-diodo para diferentes niveles de irradiación a 25°C

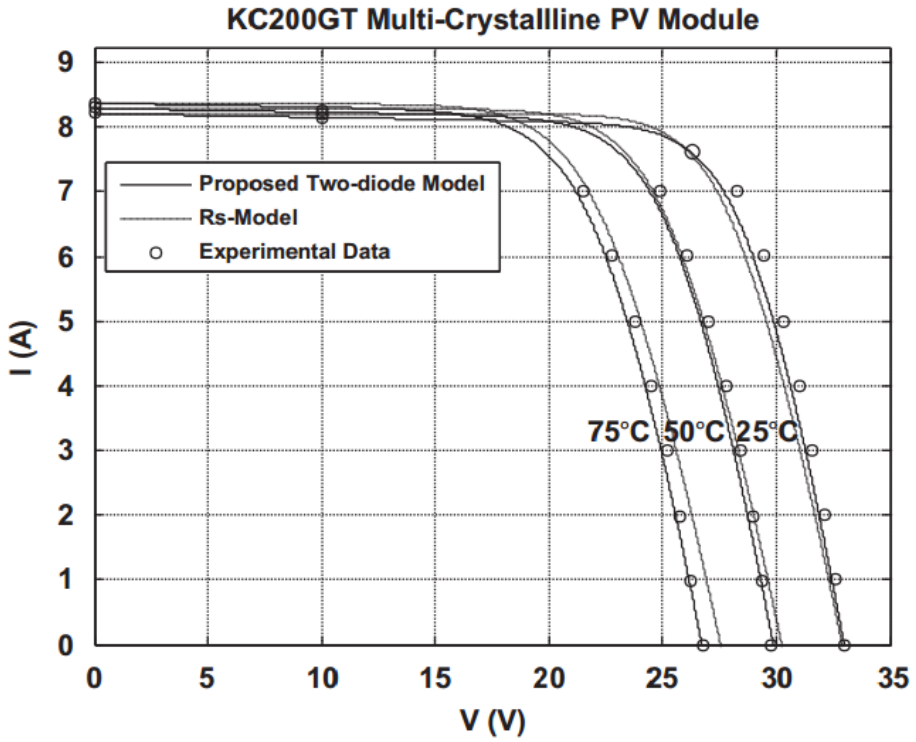


Figura 2.8 - Curvas I- V modelo de doble-diodo vs mejorado a diferentes niveles de temperatura a 1KW/m2

El valor inicial de R_p es el valor de la pendiente del segmento de línea entre el punto de corto circuito y el de máxima potencia. Para cada una de las iteraciones, el valor de R_p es calculado simultáneamente. Ya teniendo disponible los 6 parámetros, la corriente de salida de la celda puede ser determinada utilizando el método estándar de Newton-Raphson.

En conclusión, los 4 parámetros requeridos del modelo pueden ser calculados, $I_{pv,}$ $I_{o,}$ R_p *y* $R_{s.}$ De estos, solo R_p *y* R_s deben ser determinados por iteración. I_{pv} *y* I_o son obtenidos analíticamente. La variable *p* puede ser cualquier valor mayor a 2.2.

3. Modelo fotovoltaico con segmentación no uniforme

En este capítulo se presentarán conceptos relacionados al procesamiento digital que se realizará sobre las curvas FV, se abarcarán conceptos desde sistemas embebidos y aproximación de polinomios. A su vez, se presentará el concepto de arreglos sistólicos, el lenguaje de programación utilizado y como esto ayuda al objetivo de este trabajo.

3.1 Sistemas embebidos

El aumento de la potencia de cómputo necesaria para algunas aplicaciones ha permitido que las técnicas de programación en paralelo y sistemas para la resolución de problemas complejos en un tiempo óptimo se hayan desarrollado en los últimos años. Las computadoras han evolucionado más allá de los microcontroladores de 8 bits que tiempo atrás predominaban en el mercado. Hoy en día, las computadoras embebidas están organizadas en multiprocesadores que pueden ejecutar millones de líneas de código, así mismo éstas se pueden realizar en tiempo real y con niveles muy bajos de potencia.

Una característica importante en los sistemas de computación de propósito general es que se separa el diseño de hardware y software, mientras que en los sistemas de cómputo embebido se puede diseñar software y hardware simultáneamente, los cuales deben trabajar conjuntamente para lograr el mejor desempeño posible.

Recientes investigaciones en co-diseño hardware/software enfatizan la importancia del diseño concurrente. Una vez que la arquitectura del sistema ha sido definida, los componentes de hardware y software pueden ser diseñados relativamente por separado.

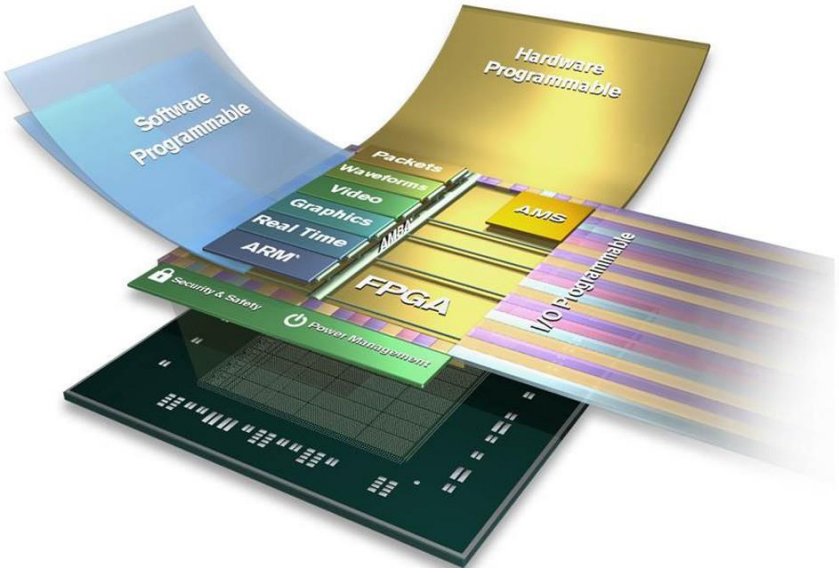


Figura 3.1 - MPSoC

Otra característica de los sistemas embebidos es la tendencia de éstos que permite la utilización de múltiples núcleos interconectados e integrados en un solo componente. Entre los dispositivos embebidos, los sistemas multiprocesadores se están convirtiendo en soluciones atractivas para aplicaciones complejas dirigidas a futuros sistemas multimedia. Los multiprocesadores con tecnología *system-on-chip* (MPSoCs por sus siglas en inglés) se componen de procesadores, memorias, co-procesadores para aplicaciones específicas y en ocasiones incluyen MPSoCs homogéneos también conocidos como procesadores especializados.

En general, las aplicaciones que se ejecutan en MPSoCs requieren un alto subsistema de memoria con suficiente ancho de banda, así como subprocesos eficientes para una mejor sincronización de los datos

3.2 Arreglos sistólicos

El término arreglo sistólico se introdujo por H. T. Kung y C. E. Leiserson (1978, 1980).

Ellos aplicaron este concepto para realizar estructuras de cómputo con una organización celular y flujo *pipeline* de datos. Aunque un número de estructuras similar y algoritmos habían sido propuestos anteriormente en los inicios de los años 60's.

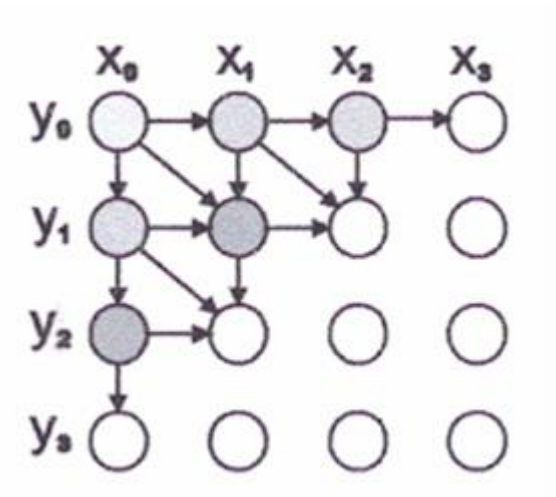


Figura 3.2 - Arreglo sistólico

El término arreglo sistólico se volvió popular en los inicios de los años 80. Se introdujo en la computación para referirse a ciertas estructuras con una organización regular celular y el flujo de datos segmentados. Los algoritmos paralelos realizados por estas estructuras se llaman algoritmos sistólicos.

Entre las propiedades de los arreglos y algoritmos sistólicos se puede mencionar su simetría, su extraordinario flujo de datos y alta eficiencia, entre otros. Esto fue sin duda, una de las razones de su rápida expansión y su amplia aceptación.

La arquitectura de arreglo sistólico formado por una o más interconexiones de un conjunto de celdas idénticas que procesan datos en una manera uniforme. Los datos que son procesados fluyen síncronamente de una celda a otra. En cada celda se realiza una operación dando un paso en el cálculo de la operación completa, en otras palabras, los datos fluyen desde una memoria externa pasando a través de varias celdas antes de que se entregue el resultado en los últimos arreglos y regresen a la memoria externa. Habiendo recibido los datos, cada celda efectúa la misma operación y transmite a la siguiente los resultados.

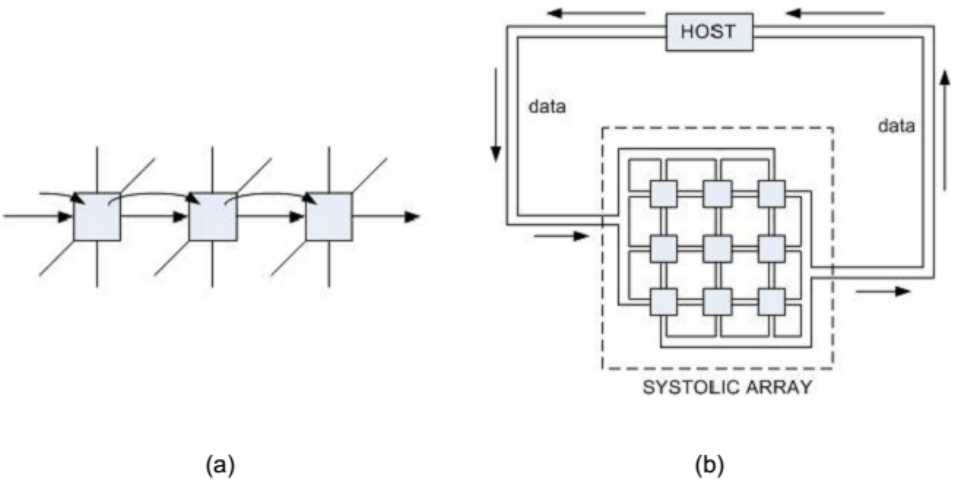


Figura 3.3 Sistema sistólico: (a) Modelo síncrono; (b) Analogía de un arreglo sistólico con el sistema circulatorio

El movimiento de los datos a través de un arreglo sistólico se asemeja al movimiento sanguíneo dentro del sistema circulatorio, como se puede apreciar en la Figura 3.3 Sistema sistólico: (a) Modelo síncrono; (b) Analogía de un arreglo sistólico con el sistema circulatorio.

Esta analogía fue la razón por la cual a estas estructuras de cómputo se les denominó sistólicas.

Este tipo de arreglos se recomienda para casos específicos entre ellos la evaluación polinomial, ya que reduce el tiempo que utilizaría un solo procesador en realizar el cálculo.

3.3 Aproximación de polinomios

La evaluación de funciones elementales es requerida en muchas aplicaciones incluyendo procesamiento digital de señales, diseño asistido por computadora, realidad virtual, y simulaciones físicas. Ejemplos de esas funciones son $\ln(x)$ y $\cos^{-1}(x)$.

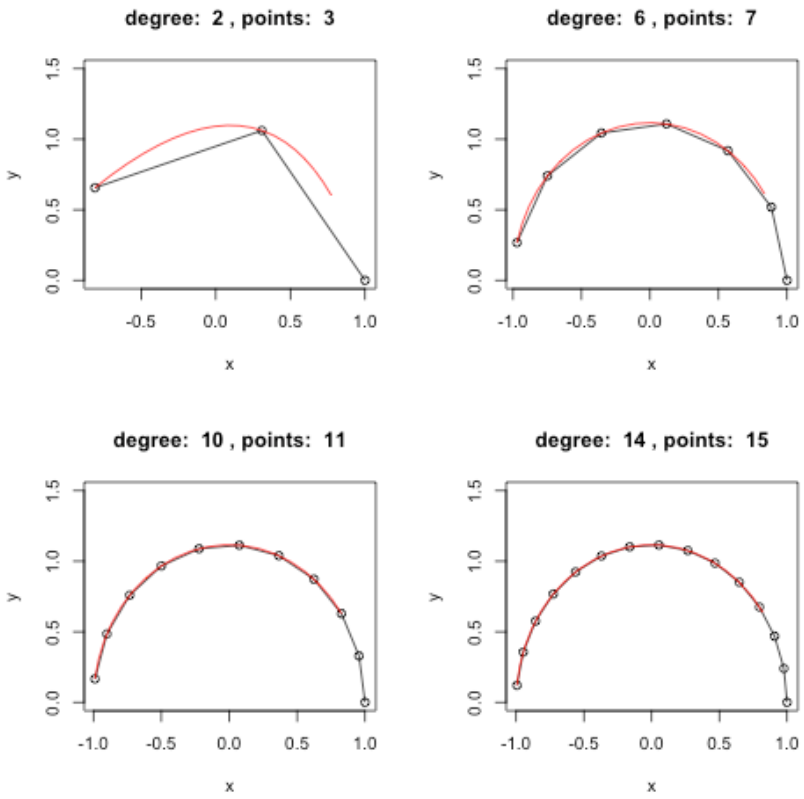


Figura 3.4 - Aproximación polinomial. Al incrementar el grado se incrementa la precisión y el número de puntos.

Desde el punto de vista algorítmico, la implementación en hardware de distintas funciones elementales puede realizarse empleando distintos métodos: iterativos y no-iterativos. Los iterativos se utilizan generalmente en aplicaciones donde precisiones arbitrarias se desean. Sin embargo, involucran altas latencias, haciéndolos poco usados en aplicaciones de alto desempeño.

Los métodos no iterativos incluyen tablas de búsqueda, aproximaciones polinomiales, y aproximaciones racionales.

En aproximación polinomial, el intervalo en que la función elemental debe ser calculada se divide en subintervalos. En cada subintervalo, la función elemental se aproxima con polinomios de bajo grado.

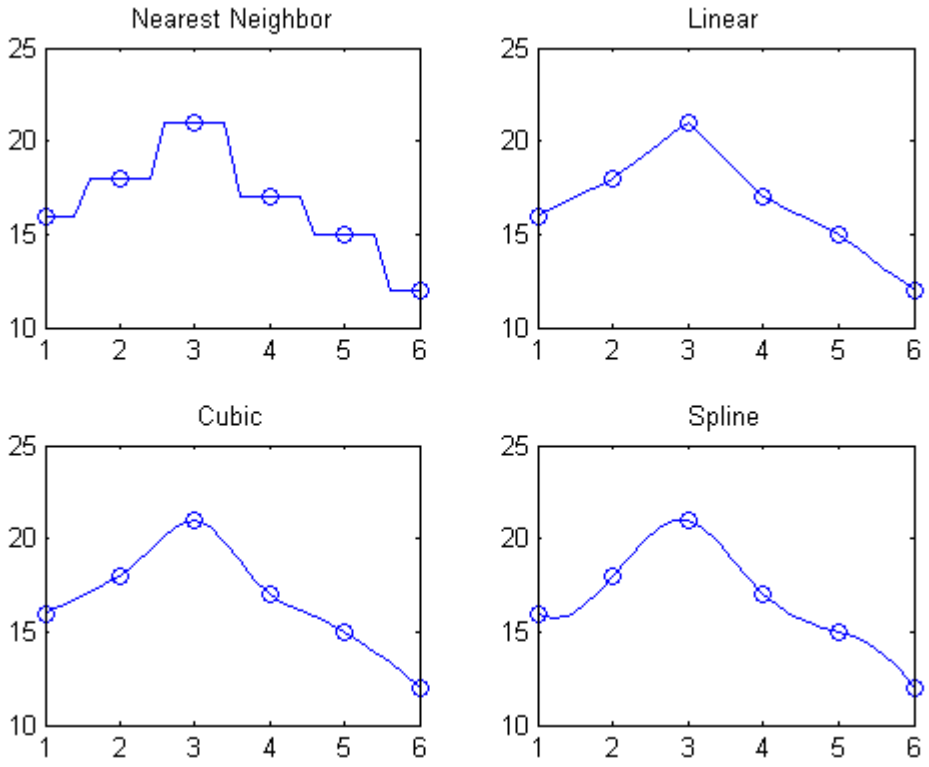


Figura 3.5 - Comparación de cuatro tipos de interpolación

La evaluación de *f(x)* consiste de tres pasos generalmente: 1) reducción de la entrada a un intervalo determinado [*a*, *b*]; 2) aproximación de la función en el rango reducido de la función; 3) reconstrucción del rango, expandiendo el resultado al rango original.

Enfocándonos en el segundo paso, se asume que *f(x)* y todas sus derivadas son continuas en el intervalo [a, b]. La aproximación se realiza primero dividiendo el intervalo [*a*, *b*] en *M* segmentos [a_k,a_{k+1}] y luego, aproximando *f(x)* en cada segmento con un polinomio de bajo grado:

$$f(x) \approx \widehat{f}(x) = \sum_{i=0}^{N} c_{k,i}(x - a_k)^i \text{ for: } a_k \leq x < a_{k+1}. \quad (5)$$

En el caso en que se considera segmentación uniforme, el número de segmentos *M* debe estar en potencias de 2, $M = 2^m$, y las dimensiones de los segmentos son iguales a: $h = \frac{b-a}{2^m}$. Esto simplifica la implementación en hardware del sistema.

Una vez que se termina de definir la segmentación, una tabla de datos que contiene los coeficientes de los polinomios se genera.

3.3.1 Segmentación no uniforme

Las aproximaciones de múltiples polinomios o “splines” son generalmente preferidas sobre las aproximaciones de polinomios simples debido a su amplio rango de ventajas en los diseños, que involucran menor uso de memoria, menor complejidad computacional y la mantención o mejora de la precisión. Tradicionalmente la aproximación por spline usa segmentación uniforme, en donde cada una de las splines cubre segmentos de igual distancia a lo largo del intervalo de la función, y se limita a potencias de dos.

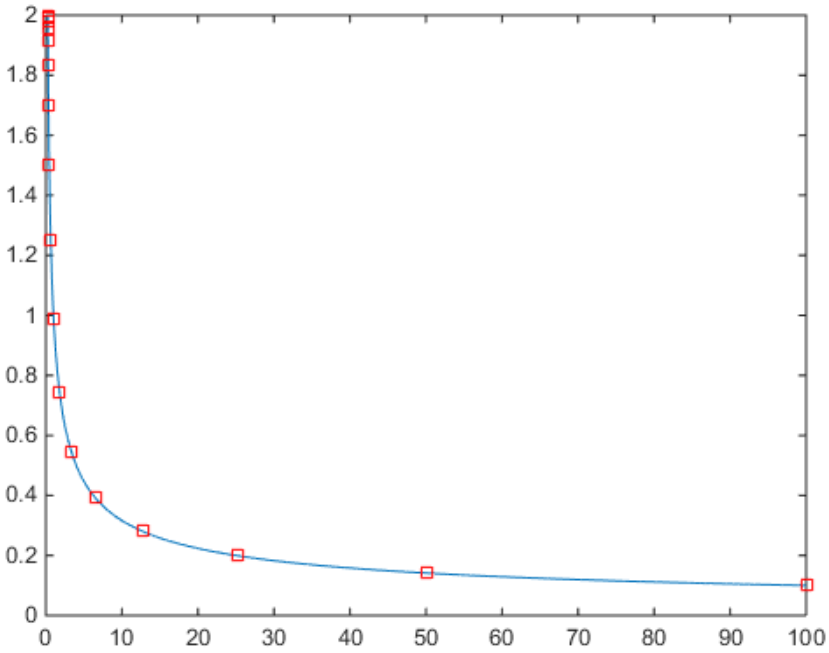


Figura 3.6 - Segmentación no uniforme aplicada a la raíz inversa

En [9] se analiza una técnica de segmentación no uniforme basada en splines, en la cual varían el tamaño en potencias de dos lográndose adaptar a las no linealidades de una función, resultando en una reducción significativa en el número de splines utilizadas con respecto a una segmentación uniforme.

En éste tipo de aproximación el intervalo completo se divide en sub-intervalos, cuando estos intervalos cuentan con la misma longitud se conoce como segmentación uniforme; la longitud de estos intervalos también puede estar determinada en potencias de dos y partir así del inicio al final, del final al inicio, o en algunos casos aumentar hasta la mitad del rango y luego ir disminuyendo; segmentación denotada por el nombre de segmentación no-uniforme.

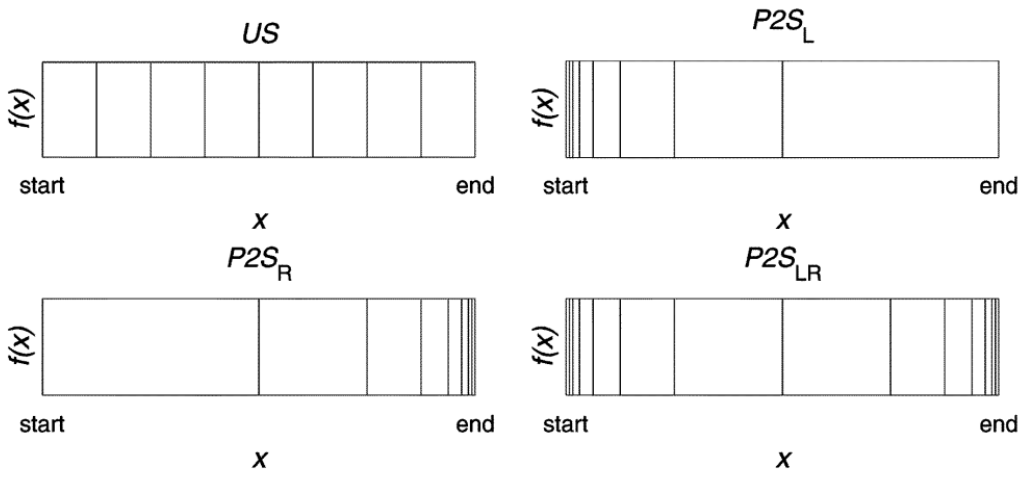


Figura 3.7 – Ilustración de Segmentación Uniforme US y tres segmentaciones no uniformes basadas en incrementos o decrementos de potencias de dos: $P2S_L$, $P2_{SR}$ y $P2S_{LR}$ [9]

Teniendo como base una segmentación no-uniforme es posible tener un segundo nivel de segmentación, teniendo potencias de dos de inicio a fin como segmentación principal y el segundo nivel consiste en una segmentación uniforme interna en cada segmento de la principal.

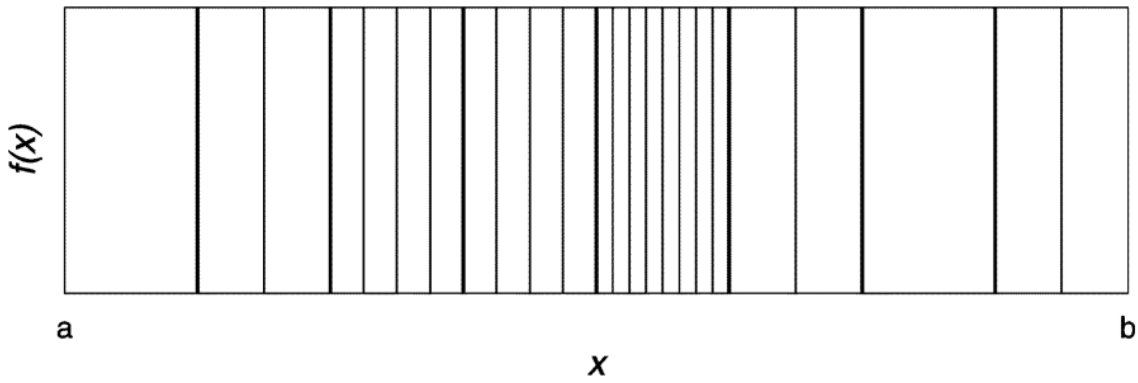


Figura 3.8 Ilustración de una segmentación de doble nivel que es uniforme en ambos niveles, se tienen 8 segmentos de igual longitud externos, y una segmentación interna que aumenta en potencias de dos. Las líneas negras indican la segmentación externa, las líneas grises la interna [9].

3.3.2 SQNR

La relación señal a ruido se utiliza en varios campos para medicar la calidad de diferentes esquemas digitales como el PCM y codecs multimedia. EL SQNR refleja la relación entre el valor máximo de señal y el error de cuantización (también conocido como ruido de cuantización) que se presenta en las conversiones analógico-digitales.

La fórmula del SQNR se deriva de la formula general del SNR y dependiendo de las señales a analizar maneja diferentes parámetros.

3.4 Field Programmable Gate Array (FPGA)

Los FPGA son dispositivos semiconductores reconfigurables que están basados alrededor de una matriz de bloques lógicos configurables (CLB – Configurable Logic Block) conectados a través de interconexiones programables. A diferencia de los circuitos integrados de aplicación específica donde el dispositivo es construido alrededor de un diseño particular, los FPGAs pueden ser programados para la aplicación deseada o los requerimientos funcionales. Se cuentan con FPGAs programables una sola vez, aunque predominan los basados en memoria SRAM los cuales puedes reprogramarse mientras el diseño va creciendo.

Los FPGAs suponen una gran ventaja al permitir al diseñador cambiar la estructura interna de los diseños aún en etapas finales, y hasta después de que la versión final haya sido manufacturada y puesta en marcha en el campo. Existen algunos FPGAs que permiten la actualización de manera completamente remota, optimizando de ésta manera los costos asociados con el re-diseño o la actualización manual de los sistemas electrónicos.

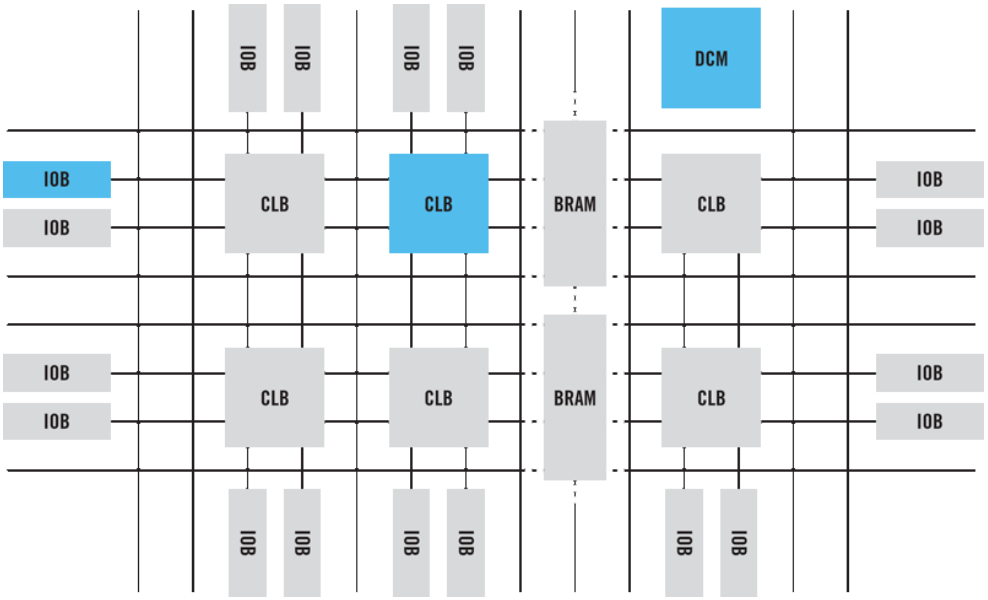


Figura 3.9 - Estructura de bloques interna del FPGA (www.xillinx.com/fpga/)

Los lenguajes de descripción de hardware (HDL, por sus siglas en inglés) y los arreglos de compuertas de campos programables (FPGA, por sus siglas en inglés) permite a los diseñadores desarrollar rápidamente y simular un circuito digital sofisticado, realizar un prototipo y verificar su operación.

Un lenguaje de descripción de hardware es una clase de lenguaje de computadora para describir formalmente un circuito electrónico digital. Este tipo de lenguaje especifica el comportamiento de un circuito electrónico, por lo general digital, el cual implementa un sistema físico.

La característica principal de un HDL es la capacidad de describir una función de un sistema pudiéndola implementar a nivel de hardware. Es posible describir la operación de un circuito, su diseño y organización, así como las pruebas de verificación para comprobar su funcionamiento. En contraste de un lenguaje de programación de software, la sintaxis y semántica de un HDL incluyen notaciones explícitas para expresar tiempo y concurrencia, las cuales son los principales atributos de hardware.

La descripción de hardware está basada en la estructura y el comportamiento del sistema que se diseña. En su estructura se representa cómo está compuesto el circuito de la arquitectura; mientras la descripción del comportamiento se representa cómo funciona la arquitectura en el módulo implementado.

Un lenguaje HDL es análogo a un lenguaje de programación de software, pero con mayores diferencias. Los lenguajes de programación de software son intrínsecamente secuenciales con limitaciones en su sintaxis y semántica. Por otro lado, los HLD pueden modelar múltiples procesos paralelos (como son flip-flops, sumadores, entre otros) que automáticamente se ejecutan independientemente uno de otro.

Ambos tipos de lenguajes son procesados por un compilador, que comúnmente se llama sintetizador en el caso de los HDL, pero cada lenguaje cuenta con diferentes objetivos. En el caso de los compiladores de software, estos convierten el código fuente en otro código que sea ejecutado por un microprocesador. Sin embargo, en los HDL el sintetizador procesa el código para transformarlo en una lista de uniones entre los componentes para después realizarlo físicamente.

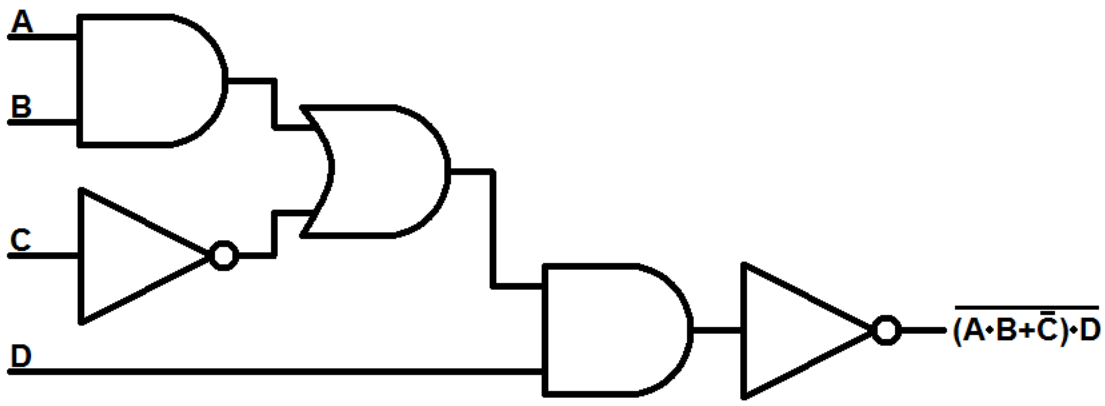


Figura 3.10 - HDL procesa el código transformándolo en una lista de uniones entre componentes

Existen numerosos HDL comerciales, de los cuales son específicos de una marca y solo se utilizan en los productos de la misma. Entre los más populares están VHDL y Verilog. Ambos lenguajes se encuentran estandarizados por la IEEE y tienen un amplio uso en la industria. Un circuito especificado en VHDL, por ejemplo, puede implementarse en diferentes tipos de chips y con herramientas CAD ofrecidas por diferentes compañías. Por una parte, VHDL ofrece una portabilidad de diseño, es decir, se centra en la funcionalidad del circuito deseado sin preocuparse mucho por los detalles de la tecnología que se usará en la implementación.

Listing 1.1 Gate-level implementation of a 1-bit comparator

```
module eq1
    // I/O ports
    (
     input wire i0, i1,
     output wire eq
    );

    // signal declaration
    wire p0, p1;

    // body
    // sum of two product terms
    assign eq = p0 | p1;
    // product terms
    assign p0 = ~i0 & ~i1;
    assign p1 = i0 & i1;

endmodule
```

Figura 3.11 - Código en Verilog

A pesar de la popularidad de VHDL, Verilog es fácil de aprender ya que cuenta con una sintaxis similar a C y Pascal. Aunque la mayoría de los conceptos de VHDL no son tan diferentes que en Verilog, VHDL es más difícil de aprender, ya que posee una sintaxis rígida influenciada por Ada, el cual es un lenguaje de programación convencional impopular.

Como se dijo, la sintaxis de Verilog es similar como la del lenguaje de programación C. Sin embargo, su semántica está basada en operaciones de hardware concurrente y es totalmente diferente de la ejecución secuencial de C.

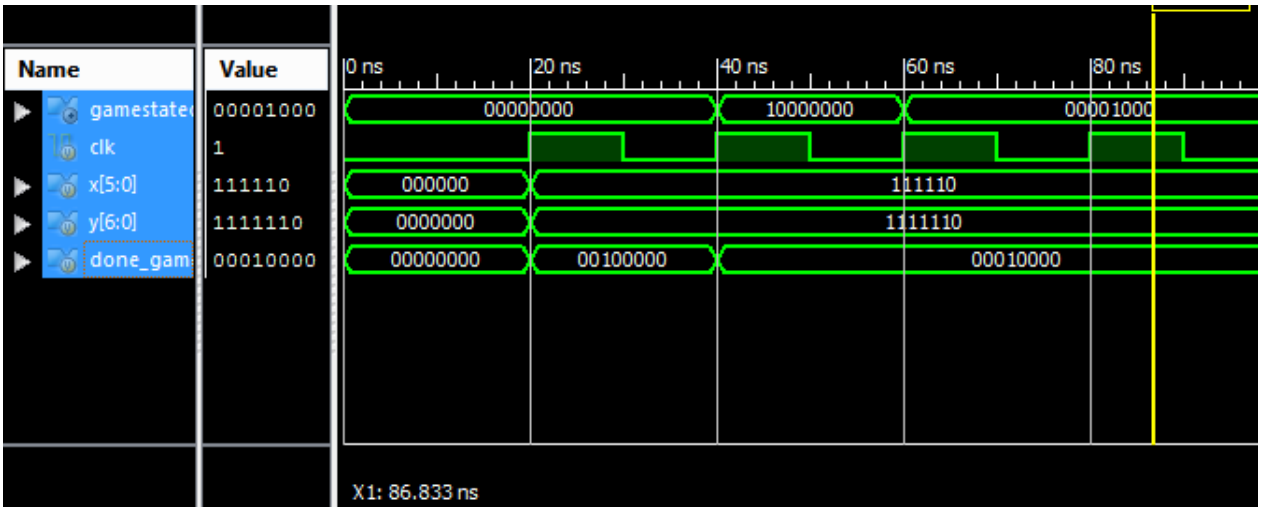
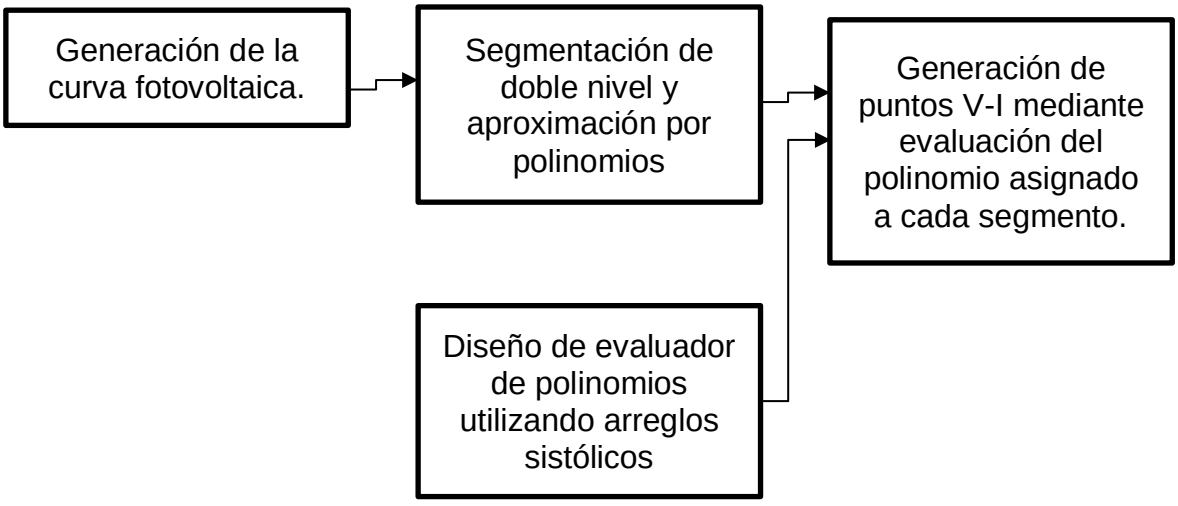


Figura 3.12 - Los resultados se analizan a través de un Testbench que indica los estados de los componentes en cada instante de tiempo

4. Hardware de un emulador fotovoltaico basado en aproximación polinomial con segmentación no uniforme

En este capítulo se explica de manera detallada la implementación del sistema del emulador fotovoltaico, empezando por la generación de la curva a aproximar, continuando con la generación de la segmentación de doble nivel y finalizando con la implementación de la misma en el fpga.

La implementación del emulador está conformada por los siguientes segmentos:



4.1 Generación de curva fotovoltaica

En “An improved Two-Diode Photovoltaic(PV) Model for PV System” [2] se establece una mejora al modelo de doble diodo, mejora que solo necesitará de 7 parámetros para generar la curva *I-V* del módulo fotovoltaico. Esta curva es generada utilizando el software Matlab, utilizando valores tomados directamente de la hoja de datos y comparando los resultados arrojados por el programa con los dados por el fabricante.

En ésta parte del proceso y con el fin de validar el modelo propuesto por [2] se realizaron diferentes pruebas con paneles fotovoltaicos de diferentes fabricantes.

4.2 Segmentación de doble nivel y aproximación por polinomios

Realizar el procesamiento de la ecuación de 7 parámetros resulta tardado y con un esfuerzo computacional complejo, utilizar la técnica de segmentación por polinomios nos permitirá aproximar con gran precisión la curva sin el esfuerzo computacional excesivo utilizando únicamente polinomios de segundo grado.

Los 7 parámetros involucrados en la aproximación son:

- I_{pv}: Corriente en el modulo fotovoltaico
- I_{o1}, I_{o2}: Corrientes de saturación inversa de los diodos 1 y 2 respectivamente.
- R_p: Resistencia en paralelo a los diodos.
- R_s: Resistencia en serie a los diodos.
- a_1, a_2: Constantes de idealidad de los diodos.

I_{o1}, I_{o2}, R_p, R_s se obtienen a través de iteración. Muchos investigaores asumen los valores de $a_1 = 1$ y $a_2 = 2$, y aunque es muy utilizada no siempre es verdadera. Muchos intentos se han realizado intentando reducir el esfuerzo computacional para el modelo de doble-diodo, sin embargo, no han tenido mucho éxito.

El código de Matlab que calcula los valores de R_p y R_s se muestra a continuación:

```
% Iterative process for Rs and Rp until Pmax,model = Pmax,experimental

while (error>tol)

%  Temperature and irradiation effect on the current

dT = T-Tn;
Ipvn = Iscn;                        % STC light-generated current
Ipv = (Ipvn + Ki*dT) *G/Gn;         % Actual light-generated current
% Isc = (Iscn + Ki*dT) *G/Gn;         % Actual short-circuit current

% Increments Rs

Rs = Rs + .01;

% Calculate Parallel resistance

Rp = Vmp*(Vmp+Imp*Rs)/(Vmp*Ipv-Vmp*Io1*exp((Vmp+Imp*Rs)/Vt/Ns/a1)+Vmp*Io1-
Vmp*Io2*exp((Vmp+Imp*Rs)/Vt/Ns/a2)+Vmp*Io2-Pmax_e);

% Solving the I-V equation for sets of (V,I) pair
clear V
clear I

V = linspace(0,val_max,L);
I = zeros(1,size(V,2));     % Current vector

for j = 1 : size(V,2) %Calculates for all voltage values
% Solves g = I - f(I,V) = 0 with Newntonn-Raphson
    g(j) = Ipv-Io1*(exp((V(j)+I(j)*Rs)/Vt/Ns/a1)-1)-
Io2*(exp((V(j)+I(j)*Rs)/Vt/Ns/a2)-1)-(V(j)+I(j)*Rs)/Rp-I(j);
    while (abs(g(j)) > 0.001)
        Id1=Io1*(exp((V(j)+I(j)*Rs)/Vt/Ns/a1)-1);
        Id2=Io2*(exp((V(j)+I(j)*Rs)/Vt/Ns/a2)-1);
        g(j) = Ipv-Id1-Id2-(V(j)+I(j)*Rs)/Rp-I(j);
        glin(j) = -Io1*Rs/Vt/Ns/a1*exp((V(j)+I(j)*Rs)/Vt/Ns/a1)-
Io2*Rs/Vt/Ns/a2*exp((V(j)+I(j)*Rs)/Vt/Ns/a2)-Rs/Rp-1;
        I(j) = I(j) - g(j)/glin(j);
    end
end % for j = 1 : size(V,2)
```

Podemos observar que a través del método de Newton-Rapson se va obteniendo el valor de cada uno de ellos hasta el punto en donde se cumpla que el punto de máxima potencia experimental sea idéntico al punto de máxima potencia del modelo, de ésta forma garantizando que ambos valores cumplirán para el modelo requerido.

Se realizó una segmentación de doble nivel sobre la curva generado por la ecuación de doble diodo de 7 parámetros, dicha segmentación consistió en una segmentación de doble nivel compuesta por una segmentación no uniforme externa de 2^2 y una uniforme interna de 2^4 teniendo como total 64 segmentos distribuidos a lo largo del intervalo.

Los límites de cada uno de los segmentos fueron calculados utilizando Matlab, los limites externos se calcularon dividiendo en dos el intervalo total a segmentar y luego dividiendo nuevamente entre dos la parte de la derecha, logrando así una segmentación en potencias de 2, lo limites internos fueron calculados dividiendo directamente la longitud del segmento externo entre el valor interno requerido.

A continuación se presenta el código en MATLAB que realiza la segmentación hibrida:

```
m = (mext)*(mint);
valor_max = 25;

%%%% Segmentación Híbrida
    boundaries=zeros(1,m);
    kronp = [1 zeros(1,mint-1)];
    limitsext(1) = 1;
    limitsext(2) = L/2;

    for z=3:mext
        limitsext(z) = (L+limitsext(z-1))/2;
    end

    limitsext(mext+1) = L;
    limits = kron(limitsext,kronp);
    limits = limits(1:end-(mint-1));

    n=0;l=0;

    for n = 1:mext;
        tamano = (limitsext(n+1)-limitsext(n))/mint;
        for l=2:mint;
            limits(((n-1)*mint)+l) = floor(limitsext(n)+ tamano*(l-1));
        end
    end
```

```
    boundaries = p0a1(limits)
    posx_values = boundaries;
    coef_ram    = zeros (length(limits)-1,d+1);
    limits (1)  = 0;

 for i=1:m
    vl = limits(i)+1;
    vu = limits(i+1);
    range = p0a1(vl:vu);
    segment = original(vl:vu);
    coef_ram(i,:) = polyfit(range,segment,d);
end

limits(1)=1;
posx_values(1)=[];

A = assigncoef(posx_values, coef_ram(:,1),m,p0a1);
B = assigncoef(posx_values, coef_ram(:,2),m,p0a1);
C = assigncoef(posx_values, coef_ram(:,3),m,p0a1);
```

Todos éstos valores se guardarán en una tabla que en el siguiente proceso nos servirá para identificar que segmento pertenece al valor y así poder saber que polinomio utilizaremos al resolver para el punto dado.

A cada uno de éstos segmentos le corresponderá una ecuación de segundo grado limitada al sub-intervalo específico, los tres coeficientes de ésta ecuación serán almacenados para su posterior utilización en el evaluador de polinomios.

4.3 Diseño de evaluador de polinomios utilizando arreglos sistólicos

La arquitectura digital del evaluador de polinomios permitirá emular en hardware diversas funciones matemáticas, por ejemplo, logarítmicas, exponenciales, trigonométricas, entre otras; obteniendo un máximo desempeño con respecto a su área y velocidad.

Este algoritmo DSP propuesto está descrito mediante la utilización de fórmulas matemáticas de alto nivel. En el diseño de esta arquitectura, las fórmulas matemáticas se necesitan convertir en elementos usando lenguajes de descripción de hardware (HDL, por sus siglas en inglés) o en representaciones gráficas. Para la realización del diseño, se utilizó el lenguaje Verilog, ya que provee de herramientas que permiten la realización de un diseño parametrizable.

La multiplicación es una operación trascendental en prácticamente todos los algoritmos de procesamiento digital de señales. En algunas aplicaciones, la mitad o más de las operaciones realizadas por un procesador involucran la multiplicación. Por lo que la presencia de un hardware multiplicador como unidad aceleradora es fundamental para la definición de un procesador DSP.

En muchos algoritmos DSP, la multiplicación está seguida por una acumulación. Por lo tanto, en muchos procesadores, el multiplicador está integrado con un sumador tal que la operación multiplicar-acumular (MAC) puede ser llevada a cabo en una instrucción por ciclo.

Project Summary | Schematic | MAC.v

C:/Users/R/Desktop/USB/P_FV_GUI2/MAC.v

```
1  `timescale 1ns / 1ps
2  module MAC
3      #( parameter I_width = 3, F_width = 4)
4      (
5          input wire signed [I_width : -F_width] p_in, x_in, coef_a,
6          output wire signed [I_width : -F_width] x_out, p_out
7      );
8
9      wire signed [2*I_width : -(2*F_width)] prod;
10     wire signed [I_width : -F_width] fixed;
11
12     assign x_out = x_in;
13     assign prod = p_in * x_in;
14     assign fixed = prod[I_width : -F_width] + prod[-(F_width+1)];
15     assign p_out = fixed + coef_a;
16
17 endmodule
```

Figura 4.1 - Código en Verilog de la MAC.

En este trabajo se realizarán operaciones de punto fijo de *n*-bits, al igual que los coeficientes de *n*-bits, se realiza un truncamiento del tamaño especificado en bits del resultado de la multiplicación y posteriormente se realiza un redondeo para obtener una mejor precisión de las operaciones aritméticas.

Para la realización de la operación redondeo a nivel bit se considera lo siguiente: Se realiza un truncamiento de acuerdo al número de bits definido, por ejemplo, si las variables son de 12 bits, el producto de la multiplicación es de 24 bits, se toman 12 bits de acuerdo al punto fijo que se utiliza y se trunca el producto. Después, se toma el siguiente dígito menos significativo y si es 1 se le suma a la variable truncada.

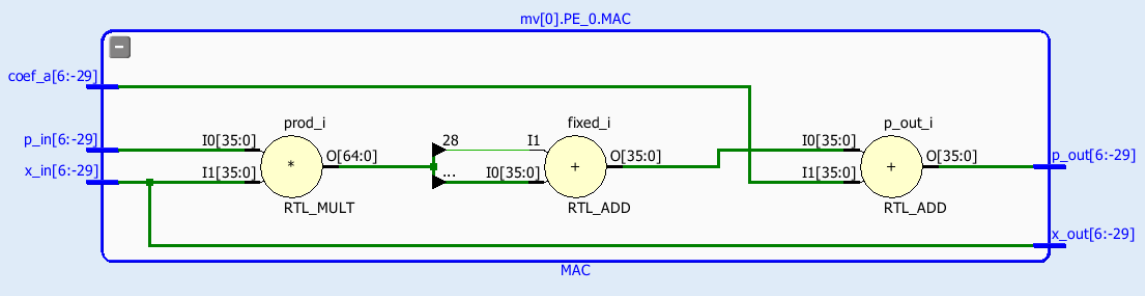
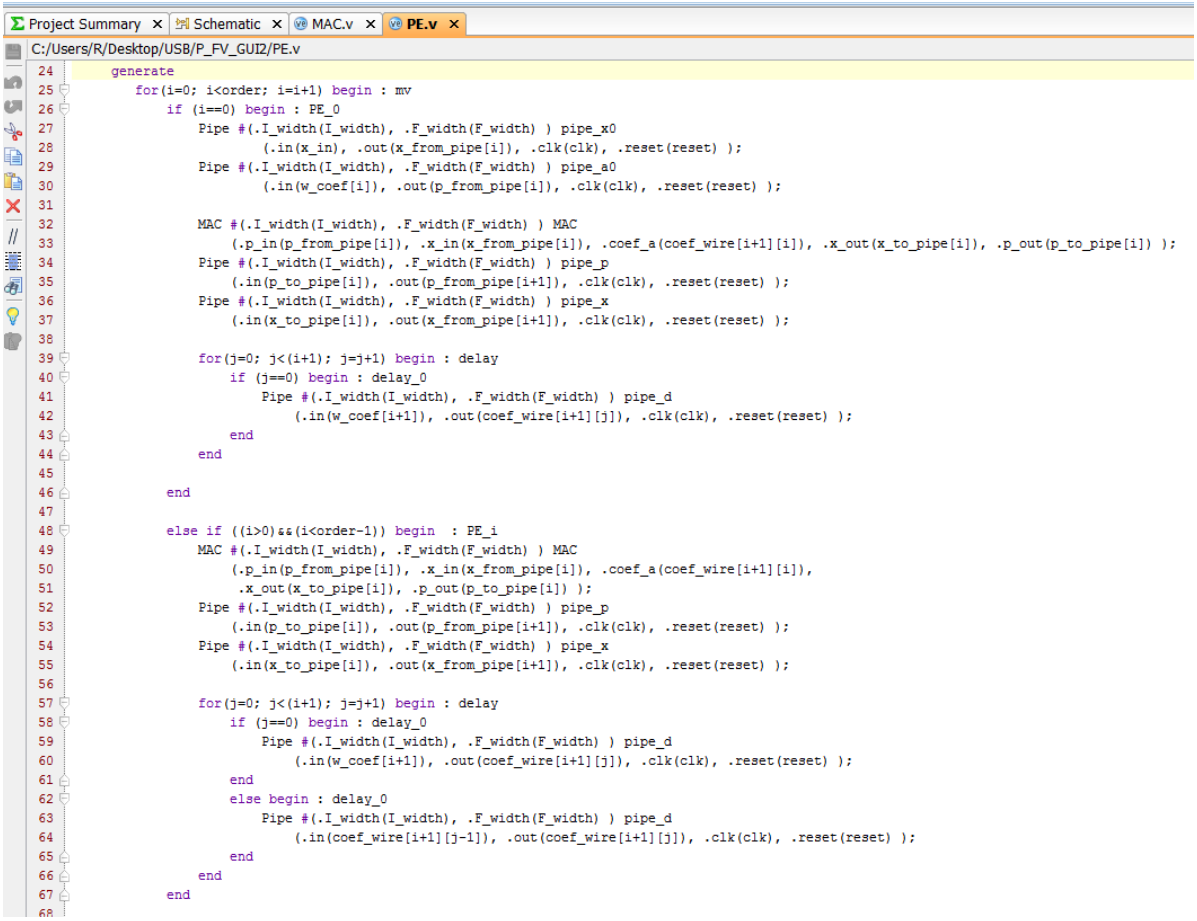


Figura 4.2 - MAC

Se implementó un multiplicador con un sumador(MAC), que en su primera instancia nos permitirá multiplicar *x*, el cuál es el valor a evaluar, por el coeficiente *a*, dando como resultado *ax* al final la siguiente etapa sumara el valor del coeficiente b teniendo al final como resultado de la MAC el término $ax + b$.

En su segunda instancia, la MAC multiplicará el término $ax + b$ por x dando como resultado $ax^2 + bx$, y la segunda etapa sumará el valor del coeficiente *c*, de ésta manera obteniendo al final el valor correspondiente a la ecuación de segundo grado $ax^2 + bx + c$.

Una vez concluida la etapa del módulo MAC, los datos fluyen hacia dos flip-flops para que en el próximo pulso de reloj ingresen en la siguiente etapa de la arquitectura evaluadora utilizando múltiples arreglos de procesadores.




```
24      generate
25          for(i=0; i<order; i=i+1) begin : mv
26              if (i==0) begin : PE_0
27                  Pipe #(.I_width(I_width), .F_width(F_width) ) pipe_x0
28                          (.in(x_in), .out(x_from_pipe[i]), .clk(clk), .reset(reset) );
29                  Pipe #(.I_width(I_width), .F_width(F_width) ) pipe_a0
30                          (.in(w_coef[i]), .out(p_from_pipe[i]), .clk(clk), .reset(reset) );
31
32                  MAC #(.I_width(I_width), .F_width(F_width) ) MAC
33                      (.p_in(p_from_pipe[i]), .x_in(x_from_pipe[i]), .coef_a(coef_wire[i+1][i]), .x_out(x_to_pipe[i]), .p_out(p_to_pipe[i]) );
34                  Pipe #(.I_width(I_width), .F_width(F_width) ) pipe_p
35                      (.in(p_to_pipe[i]), .out(p_from_pipe[i+1]), .clk(clk), .reset(reset) );
36                  Pipe #(.I_width(I_width), .F_width(F_width) ) pipe_x
37                      (.in(x_to_pipe[i]), .out(x_from_pipe[i+1]), .clk(clk), .reset(reset) );
38
39                  for(j=0; j<(i+1); j=j+1) begin : delay
40                      if (j==0) begin : delay_0
41                          Pipe #(.I_width(I_width), .F_width(F_width) ) pipe_d
42                              (.in(w_coef[i+1]), .out(coef_wire[i+1][j]), .clk(clk), .reset(reset) );
43                      end
44                  end
45
46              end
47
48              else if ((i>0)&&(i<order-1)) begin  : PE_i
49                  MAC #(.I_width(I_width), .F_width(F_width) ) MAC
50                      (.p_in(p_from_pipe[i]), .x_in(x_from_pipe[i]), .coef_a(coef_wire[i+1][i]),
51                       .x_out(x_to_pipe[i]), .p_out(p_to_pipe[i]) );
52                  Pipe #(.I_width(I_width), .F_width(F_width) ) pipe_p
53                      (.in(p_to_pipe[i]), .out(p_from_pipe[i+1]), .clk(clk), .reset(reset) );
54                  Pipe #(.I_width(I_width), .F_width(F_width) ) pipe_x
55                      (.in(x_to_pipe[i]), .out(x_from_pipe[i+1]), .clk(clk), .reset(reset) );
56
57                  for(j=0; j<(i+1); j=j+1) begin : delay
58                      if (j==0) begin : delay_0
59                          Pipe #(.I_width(I_width), .F_width(F_width) ) pipe_d
60                              (.in(w_coef[i+1]), .out(coef_wire[i+1][j]), .clk(clk), .reset(reset) );
61                      end
62                      else begin : delay_0
63                          Pipe #(.I_width(I_width), .F_width(F_width) ) pipe_d
64                              (.in(coef_wire[i+1][j-1]), .out(coef_wire[i+1][j]), .clk(clk), .reset(reset) );
65                      end
66                  end
67              end
68
```

Figura 4.3 – Pipelines Verilog MAC

El pipeline es una técnica utilizada para la reducción de caminos críticos en un algoritmo digital. Con esta técnica se puede explotar el incremento de la velocidad de reloj o la reducción del consumo de potencia a una misma velocidad. Los caminos críticos son el mínimo tiempo que se requiere para el procesamiento de una muestra.

Dicho de otra manera, una estructura pipeline funciona como una tubería en la que los datos fluyen a través de la misma y en la que cada elemento procesador está separado por flip-flops. Estos permiten que cada elemento procesador termine en un periodo de reloj todas las operaciones lógicas que se transferirán al siguiente bloque de manera síncrona y produciendo un periodo de latencia. La latencia se define como el periodo de tiempo que tarda en salir el primer dato de un sistema pipeline o de un sistema secuencial.

Con el fin de apreciar la estructura del evaluador de polinomios se agruparán en un bloque las estructuras del módulo MAC y los flip-flops, el cual se le denominará como *PE_module*.

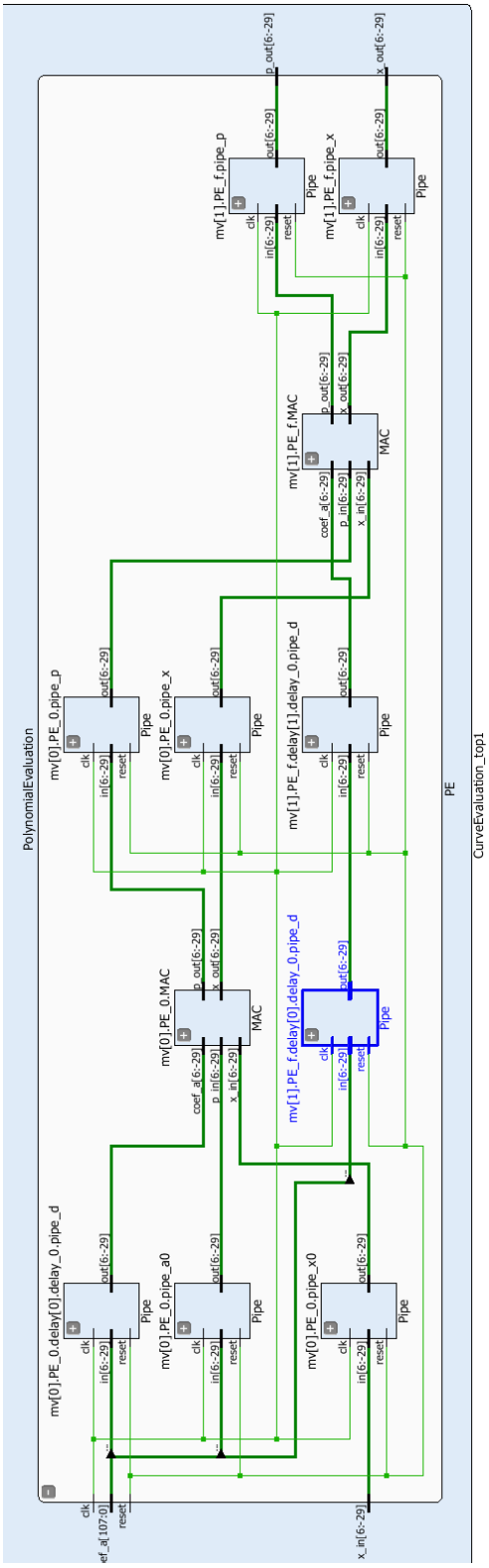


Figura 4.4 Evaluador de polinomios completo.

De manera similar al proceso realizado en Matlab se obtienen los valores del Diodo 1, Diodo 2 y el valor de la resistencia y al final se suman en conjunto con *Ipv*, obteniendo el valor de corriente *I* para un valor de voltaje *V* dado.

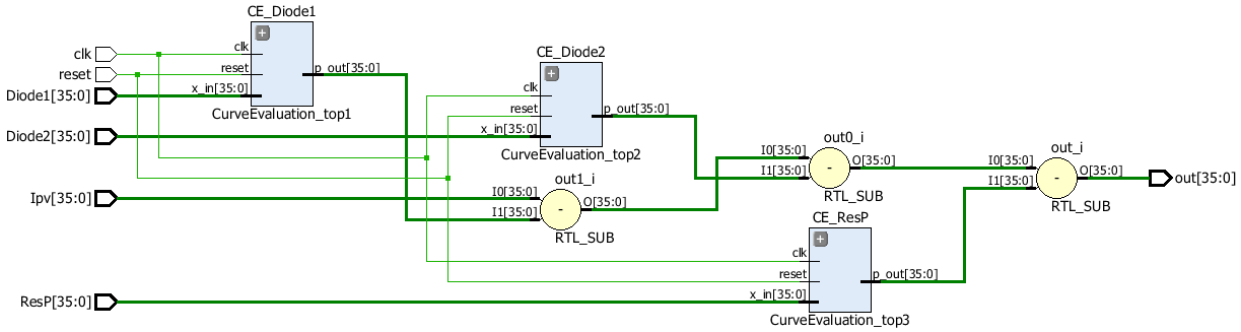


Figura 4.5 Arquitectura sistema evaluador fotovoltaico

5. Análisis de Resultados

En este capítulo se presentarán los resultados que se obtuvieron en cada una de las secciones descritas en el capítulo anterior. Se realizará la validación de la ecuación de doble diodo contrastando los resultados arrojados por MATLAB con los encontrados en las hojas de datos. Se expondrán los resultados entregados por el evaluador de polinomios, así como un análisis del máximo error posible para cada una de las evaluaciones. Por último, se presentarán los resultados obtenidos por la aproximación de doble nivel utilizando la medición del SQNR como referencia.

5.1 Generación de curva fotovoltaica

Mediante el uso de MATLAB se replicaron los resultados obtenidos por [2], los cuales utilizan la ecuación de doble-diodo, se contrastan los datos arrojados con los encontrados en la hoja de datos para validar la precisión de utilizar los 7 parámetros que pide la ecuación.

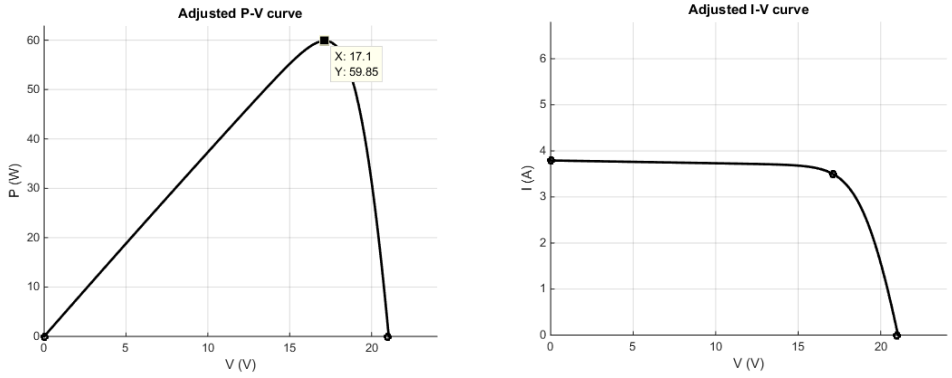
5.1.1 Casos de estudio

Se analizarán distintos casos de panel fotovoltaico a los cuáles se les aplicará el método propuesto para validar la ecuación original y utilizar éstos valores como base para la aproximación.

5.1.1.1 Celda MSX-60

Datos de modelo utilizados obtenidos de la hoja de datos:

Rp_min = 55.857143
Rp = 164.585828
Rs_max = 1.142857
Rs = 0.340000
a1 = 1.000000
a2 = 1.200000
T = 25.000000
G = 1000.000000
Pmax,m = 59.850000 (model)
Pmax,e = 59.850000 (experimental)
tol = 0.001000
P_error = -0.000000
Ipv = 3.800000
Isc = 3.791560
Voc = 21.000000
Vmp = 17.100000
Imp = 3.500000
Io1=Io2 = 4.703980e-10



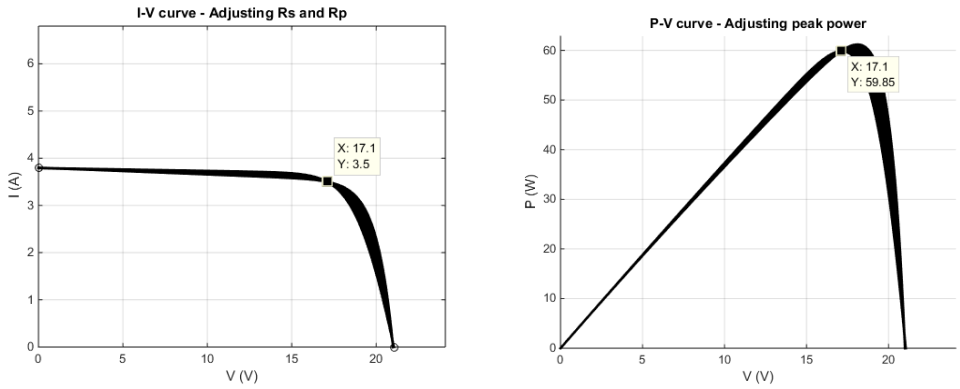


Figura 5.1 – Graficas en MATLAB modelación de panel fotovoltaico MSX-60

Se realizó una comparación entre los datos arrojados por Matlab y la hoja de datos del fabricante.

Tabla 5-1 Comparación valores MSX-60 hoja de datos vs ecuación de doble diodo MATLAB

Característica	MATLAB	Hoja de datos
Typical peak power	59.85 W	60 W
Voltage @ peak power	17.1 V	17.1 V
Current @ peak power	3.5A	3.5 A
Short-circuit current (Isc)	3.8 A	3.8A
Open-circuit voltaje (Voc)	21.1V	21.1V

En base a los datos recopilados se validó el modelo de doble diodo utilizando los 7 parámetros propuestos por Ishaque [2], el error entre la hoja de datos y el modelo propuesto es menor al 1% permitiendo utilizar ésta ecuación como base de la aproximación por polinomios.

5.1.1.2 Celda Kyocera KG200GT

```
%% Information from the Kyocera KG200GT module datasheet

Iscn = 8.2;             %STC short-circuit voltage (A)
Vocn = 32.9;            %STC array open-circuit voltage (V)
Imp = 7.61;              %PV Module current @ maximum power point (A)
Vmp = 26.3;             %PV Module voltage @ maximum power point (V)
Pmax_e = Vmp*Imp;       %PV Module maximum output peak power (W)
Kv = -123e-3;            %Voltage/temperature coefficient (V/K)
Ki = .00318;              %Current/temperature coefficient (A/K)
Ns = 54;                %Nunber of series cells in a PV Module

Model information:
                     Rp_min = 43.708991
                         Rp = 169.430681
                     Rs_max = 0.867280
                         Rs = 0.320000
                          a1 = 1.000000
                          a2 = 1.200000
                          T = 25.000000
                          G = 1000.000000
                     Pmax,m = 200.143320  (model)
                     Pmax,e = 200.143000  (experimental)
                        tol = 0.001000
                    P_error = 0.000320
                        Ipv = 8.200000
                        Isc = 8.183953
                        Voc = 32.800000
                        Vmp = 26.400000
                        Imp = 7.581186
                       Io1=Io2 = 4.122982e-10
```

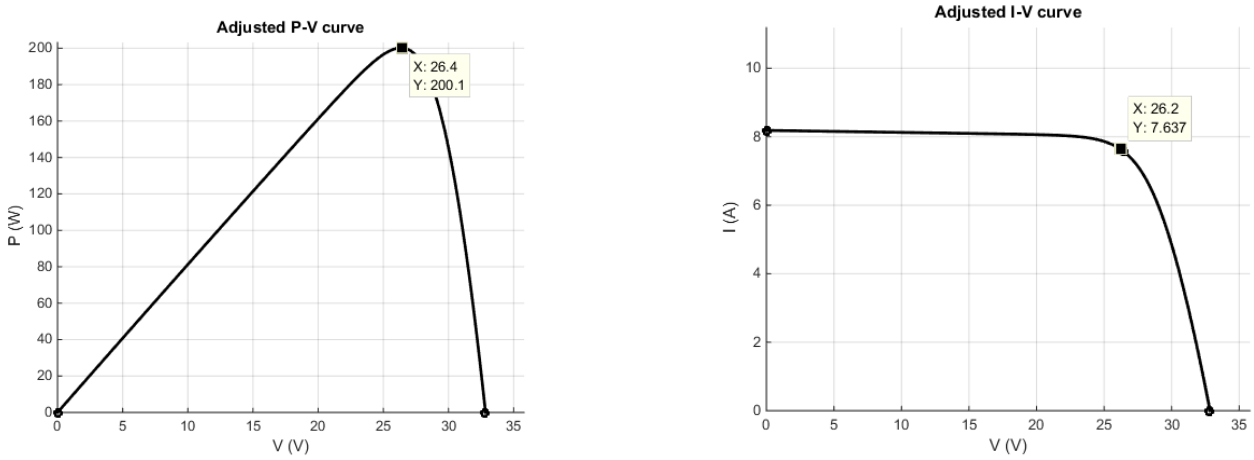


Figura 5.2 - Panel FV Kyocera KG200GT

Se realizó una comparación entre los datos arrojados por Matlab y la hoja de datos del fabricante.

Característica	MATLAB	Hoja de datos
Typical peak power	200.1433220 W	200.143000 W
Voltage @ peak power	26.4 V	26.3 V
Current @ peak power	7.5811 A	7.61 A
Short-circuit current (Isc)	8.183953 A	8.2 A
Open-circuit voltage (Voc)	32.8 V	32.9 V

En base a los datos recopilados se validó el modelo de doble diodo utilizando los 7 parámetros propuestos por Ishaque [2], el error entre la hoja de datos y el modelo propuesto es menor al 1% permitiendo utilizar ésta ecuación como base de la aproximación por polinomios.

5.1.1.3 SQ150-PC

```
%% Information from the SQ150-PC module datasheet

Iscn = 4.8;              %STC short-circuit voltage (A)
Vocn = 43.4;             %STC array open-circuit voltage (V)
Imp = 4.4;               %PV Module current @ maximum power point (A)
Vmp = 34;              %PV Module voltage @ maximum power point (V)
Pmax_e = Vmp*Imp;        %PV Module maximum output peak power (W)
Kv = -161e-3;             %Voltage/temperature coefficient (V/K)
Ki = .0014;               %Current/temperature coefficient (A/K)
Ns = 72;                 %Nunber of series cells in a PV Module
```

Model information:

```
Rp_min = 82.863636
    Rp = 274.792111
Rs_max = 2.136364
    Rs = 0.900000
     a1 = 1.000000
     a2 = 1.200000
     T = 25.000000
     G = 1000.000000
Pmax,m = 149.600000  (model)
Pmax,e = 149.600000  (experimental)
   tol = 0.001000
P_error = -0.000000
    Ipv = 4.800000
    Isc = 4.783968
    Voc = 43.300000
    Vmp = 34.000000
    Imp = 4.400000
    Io1=Io2 = 3.105979e-10
```

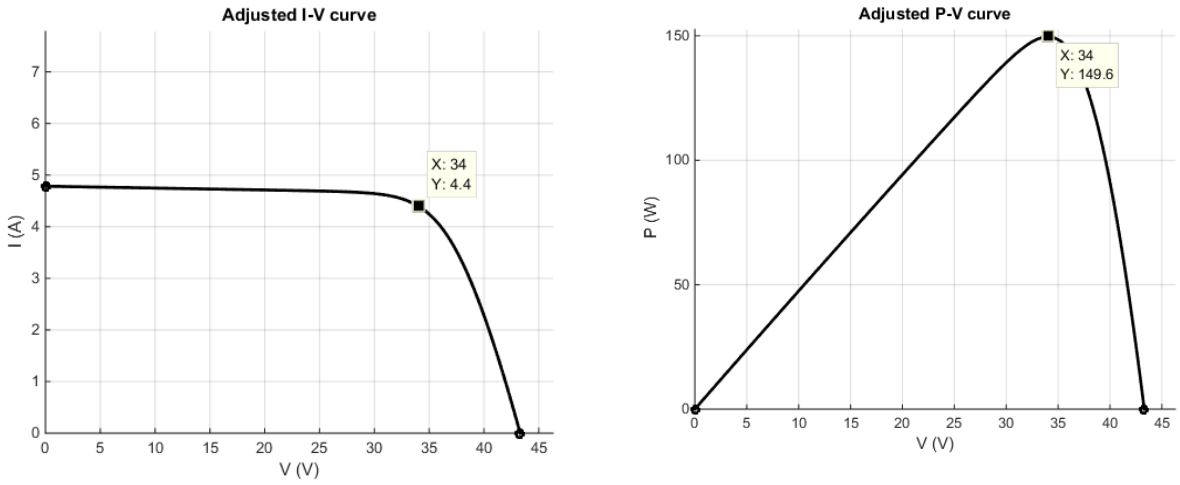


Figura 5.3 - Modulo FV SQ150-PC

Se realizó una comparación entre los datos arrojados por Matlab y la hoja de datos del fabricante.

Característica	MATLAB	Hoja de datos
Typical peak power	149.600 W	149.600 W
Voltage @ peak power	34 V	34 V
Current @ peak power	4.4 A	4.4 A
Short-circuit current (Isc)	4.783968 A	4.8 A
Open-circuit voltage (Voc)	43.3 V	43.4 V

En base a los datos recopilados se validó el modelo de doble diodo utilizando los 7 parámetros propuestos por Ishaque [2], el error entre la hoja de datos y el modelo propuesto es menor al 1% permitiendo utilizar ésta ecuación como base de la aproximación por polinomios.

5.1.1.4 Shell SP-70

```
%% Information from the Shell SP-70 module datasheet

Iscn = 4.7;             %STC short-circuit voltage (A)
Vocn = 21.4;            %STC array open-circuit voltage (V)
Imp = 4.25;              %PV Module current @ maximum power point (A)
Vmp = 16.5;             %PV Module voltage @ maximum power point (V)
Pmax_e = Vmp*Imp;       %PV Module maximum output peak power (W)
Kv = -76e-3;            %Voltage/temperature coefficient (V/K)
Ki = .002;              %Current/temperature coefficient (A/K)
Ns = 36;                %Nunber of series cells in a PV Module

Model information:
                          Rp_min = 35.513725
                              Rp = 94.964275
                          Rs_max = 1.152941
                              Rs = 0.510000
                               a1 = 1.000000
                               a2 = 1.200000
                               T = 25.000000
                               G = 1000.000000
                          Pmax,m = 70.125000  (model)
                          Pmax,e = 70.125000  (experimental)
                             tol = 0.001000
                         P_error = 0.000000
                             Ipv = 4.700000
                             Isc = 4.673846
                             Voc = 21.300000
                             Vmp = 16.500000
                             Imp = 4.250000
                             Io1=Io2 = 4.206466e-10
```

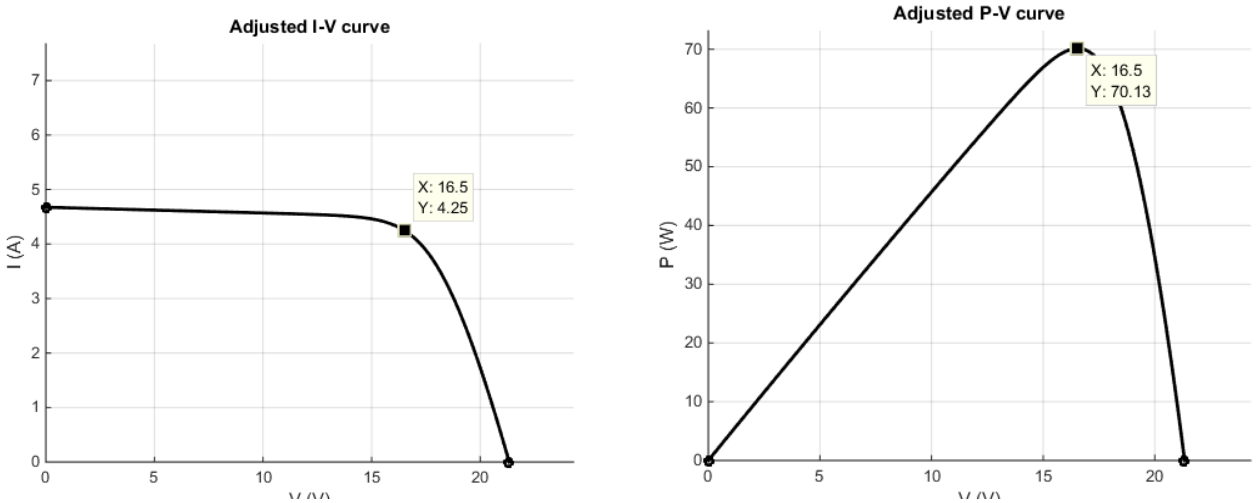


Figura 5.4 – Shell SP-70

Se realizó una comparación entre los datos arrojados por Matlab y la hoja de datos del fabricante.

Característica	MATLAB	Hoja de datos
Typical peak power	70.125 W	70.125 W
Voltage @ peak power	16.5 V	16.5 V
Current @ peak power	4.25 A	4.25 A
Short-circuit current (Isc)	4.673846 A	4.7 A
Open-circuit voltage (Voc)	21.3 V	21.4 V

En base a los datos recopilados se validó el modelo de doble diodo utilizando los 7 parámetros propuestos por Ishaque [2], el error entre la hoja de datos y el modelo propuesto es menor al 1% permitiendo utilizar ésta ecuación como base de la aproximación por polinomios.

5.1.1.5 Shell ST ST40

```
%% Information from the Shell ST ST40 module datasheet

%  You may change these parameters to fit the I-V model
%  to other types of PV module.

Iscn = 2.68;             %STC short-circuit voltage (A)
Vocn = 23.3;            %STC array open-circuit voltage (V)
Imp = 2.41;              %PV Module current @ maximum power point (A)
Vmp = 16.6;             %PV Module voltage @ maximum power point (V)
Pmax_e = Vmp*Imp;       %PV Module maximum output peak power (W)
Kv = -100e-3;            %Voltage/temperature coefficient (V/K)
Ki = .00035;              %Current/temperature coefficient (A/K)
Ns = 36;                %Nunber of series cells in a PV Module

Model information:
                     Rp_min = 58.701398
                         Rp = 204.849192
                     Rs_max = 2.780083
                         Rs = 1.710000
                          a1 = 1.000000
                          a2 = 1.200000
                          T = 25.000000
                          G = 1000.000000
                     Pmax,m = 40.006000  (model)
                     Pmax,e = 40.006000  (experimental)
                        tol = 0.001000
                    P_error = -0.000000
                        Ipv = 2.680000
                        Isc = 2.657329
                        Voc = 23.200000
                        Vmp = 16.600000
                        Imp = 2.410000
                        Io1=Io2 = 3.074866e-11
```

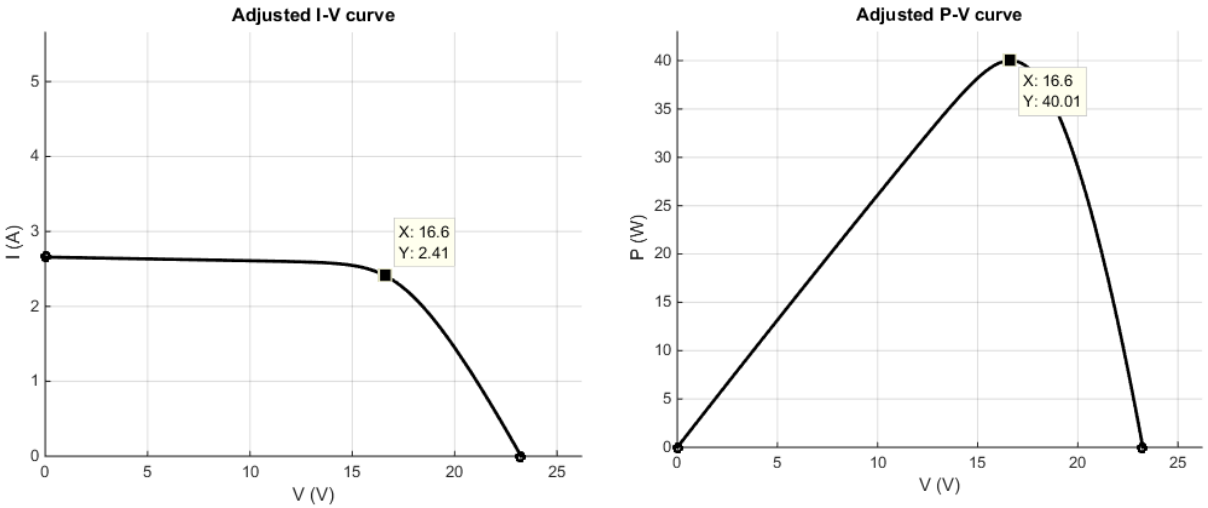


Figura 5.5 - Shell ST ST40

Se realizó una comparación entre los datos arrojados por Matlab y la hoja de datos del fabricante.

Característica	MATLAB	Hoja de datos
Typical peak power	40.006 W	40.006 W
Voltage @ peak power	16.6 V	16.6 V
Current @ peak power	2.41 A	2.41 A
Short-circuit current (Isc)	2.657329 A	2.68 A
Open-circuit voltage (Voc)	23.2 V	23.3 V

En base a los datos recopilados se validó el modelo de doble diodo utilizando los 7 parámetros propuestos por Ishaque [2], el error entre la hoja de datos y el modelo propuesto es menor al 1% permitiendo utilizar ésta ecuación como base de la aproximación por polinomios.

5.2 Segmentación de doble nivel y aproximación por polinomios

Se realizó una segmentación de doble nivel sobre la curva generado por la ecuación de doble diodo de 7 parámetros, dicha segmentación consistió en una segmentación de doble nivel compuesta por una segmentación no uniforme externa de 2^2 y una uniforme interna de 2^4 teniendo como total 64 segmentos distribuidos a lo largo del intervalo.

Los límites de cada uno de los segmentos fueron calculados utilizando Matlab, los limites externos se calcularon dividiendo en dos el intervalo total a segmentar y luego dividiendo nuevamente entre dos la parte de la derecha, logrando así una segmentación en potencias de 2, lo limites internos fueron calculados dividiendo directamente la longitud del segmento externo entre el valor interno requerido. Todos éstos valores se guardarán en una tabla que en el siguiente proceso nos servirá para identificar que segmento pertenece al valor y así poder saber que polinomio utilizaremos al resolver para el punto dado.

coef_ram

64x3 double

	1	2	3
30	-0.1153	3.8485	-28.8916
31	-0.1549	5.2702	-41.6727
32	-0.1984	6.8673	-56.3391
33	-0.2303	8.0633	-67.5308
34	-0.2496	8.7940	-74.4544
35	-0.2662	9.4313	-80.5545
36	-0.2795	9.9445	-85.5164
37	-0.2889	10.3107	-89.0932
38	-0.2941	10.5176	-91.1329
39	-0.2953	10.5635	-91.5896
40	-0.2926	10.4572	-90.5185
41	-0.2867	10.2152	-88.0588
42	-0.2780	9.8593	-84.4070
43	-0.2673	9.4135	-79.7903
44	-0.2707	9.5584	-81.3387
45	-0.2419	8.3466	-68.5829
46	-0.2283	7.7669	-62.4095
47	-0.2146	7.1787	-56.0880

Figura 5.6 - Coeficientes generados en la aproximación por polinomios segmentación de doble nivel

Una vez obtenidos los límites de cada uno de los segmentos y sabiendo el polinomio que le corresponde a cada uno podemos encontrar el valor de aproximación para cualquier punto requerido dentro del intervalo de la ecuación.

De ésta manera logramos obtener cualquier pareja de puntos *I – V* sin tener que recurrir a la ecuación de doble diodo que implica mayor esfuerzo computacional.

Para tener una mayor facilidad en el manejo de los datos se optó por una arquitectura de punto fijo, al estar trabajando en Matlab con punto flotante debemos realizar el procesamiento de los datos de tal manera que queden en punto fijo y poder realizar el análisis del error de manera adecuada. Esto se logró utilizando la función *fi* en Matlab, la cual nos arrojará los valores de coeficientes en punto fijo.

Segmento	Punto flotante	Punto fijo
1	-1.08011012478052e-09	0
2	-2.51148176194629e-09	0
3	-5.83968372827481e-09	0
4	-1.35784324613696e-08	0
5	-3.15724702232219e-08	0
6	-7.34121377872950e-08	0
7	-1.70697411768919e-07	0
8	-3.96904416355904e-07	0
9	-9.22878704437362e-07	0
10	-2.14586655912964e-06	0
11	-4.98952667047909e-06	0
12	-1.16014592752147e-05	-1.52587890625000e-05
13	-2.69747895635521e-05	-3.05175781250000e-05
14	-6.27170053305030e-05	-6.10351562500000e-05
15	-0.000145804273724554	-0.000152587890625000

Figura 5.7 - Comparación Punto fijo vs Punto flotante

Esta operación sobre los datos de la aproximación generará un error que aumentará el error de los valores aproximados respecto a los originales obtenidos por la ecuación de doble diodo.

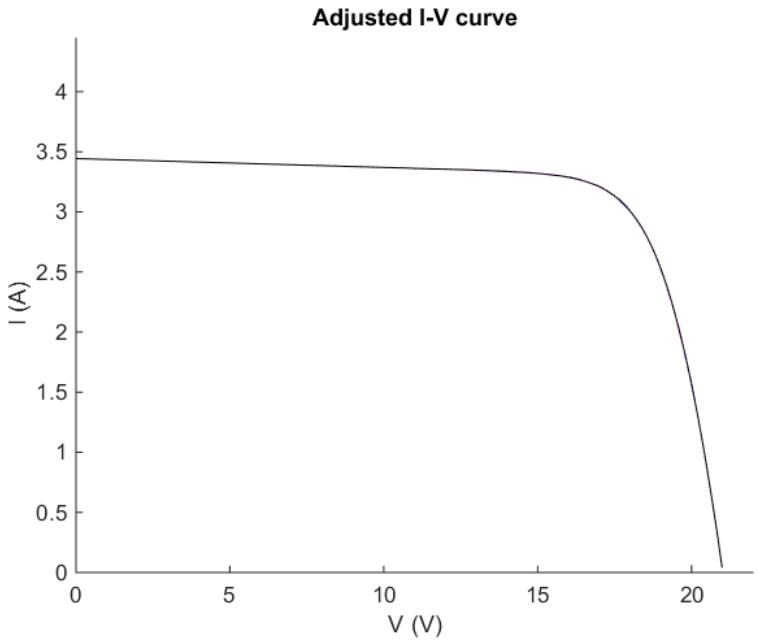


Figura 5.8 - Vista de la aproximación de doble nivel en el intervalo de 0 a 25. [Rojo - Original; Negro - Aproximación punto fijo; Rosa - Aproximación punto flotante]

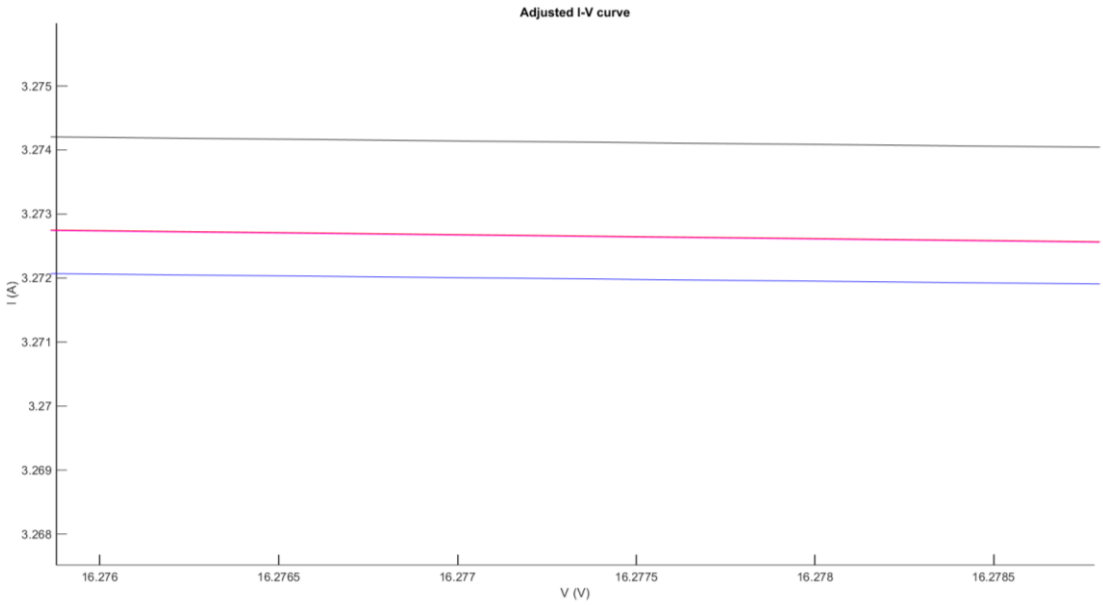


Figura 5.9 - Detalle aproximación de doble nivel. [Rojo – Original ecuación de doble diodo; Negro - Aproximación punto fijo; Azul - Aproximación punto flotante]

Utilizando como parámetro de comparación el SQNR encontrando que para los valores de:

Segmentación externa: $\mathbf{2^2}$ Segmentación interna: $\mathbf{2^4}$

Se obtuvo un valor de SQNR de: **71.170470**

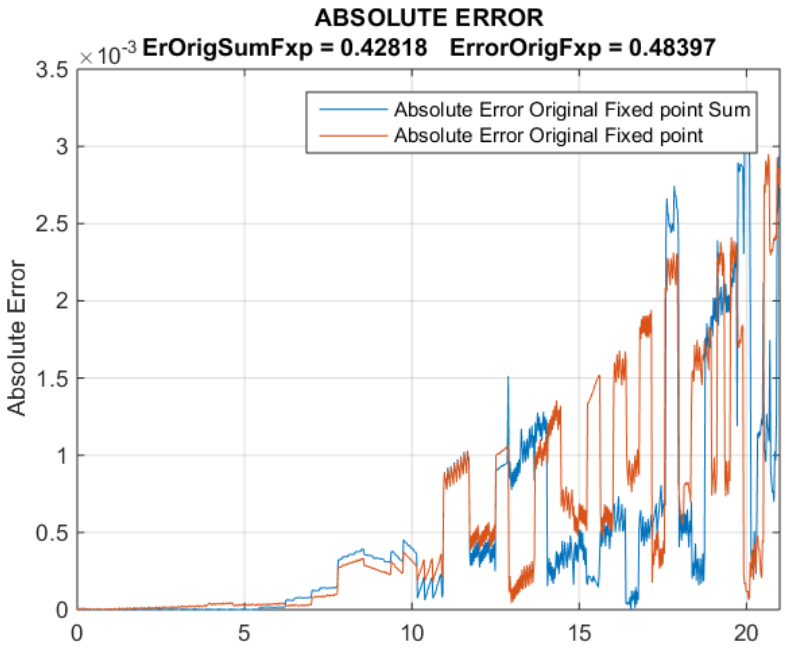


Figura 5.10 - Error absoluto punto fijo

Se realizó también la gráfica del error absoluto respecto a la gráfica de ecuación original, obteniendo como resultado que el máximo error a lo largo de toda la curva es de 2.9 x 10^{-3}.

Tomando en cuenta el valor máximo de la curva en 3.44 y el máximo error, tenemos un error máximo porcentual de: 0.084%

5.2.1 Casos de estudio

Se tomarán como referencia 6 modelos de panel fotovoltaico cuyas características serán obtenidas directamente desde la hoja de datos. Ésta muestra contendrá 2 celdas multi-cristalinas, 3 celdas mono-cristalinas y 1 thin film

Se expondrá la matriz de coeficientes generadas para cada una de ellas, análisis del error y valor del SQNR final arrojado para validar el método propuesto.

5.2.1.1 Celda MSX-60

Se obtuvieron 64 valores para cada uno de los coeficientes de la ecuación de segundo grado. Se muestran en la imagen los últimos 20 valores a manera de referencia.

coef_ram | coef_ramA_fxp_obj

64x3 double

	1	2	3
40	-0.2926	10.4572	-90.5185
41	-0.2867	10.2152	-88.0588
42	-0.2780	9.8593	-84.4070
43	-0.2673	9.4135	-79.7903
44	-0.2707	9.5584	-81.3387
45	-0.2419	8.3466	-68.5829
46	-0.2283	7.7669	-62.4095
47	-0.2146	7.1787	-56.0880
48	-0.2011	6.5945	-49.7536
49	-0.1881	6.0238	-43.5087
50	-0.1756	5.4732	-37.4305
51	-0.1638	4.9473	-31.5735
52	-0.1527	4.4487	-25.9714
53	-0.1423	3.9788	-20.6462
54	-0.1327	3.5379	-15.6068
55	-0.1238	3.1256	-10.8542
56	-0.1155	2.7411	-6.3834
57	-0.1078	2.3830	-2.1854
58	-0.1008	2.0500	1.7511
59	-0.0943	1.7405	5.4402
60	-0.0883	1.4529	8.8958
61	-0.0828	1.1857	12.1326
62	-0.0777	0.9374	15.1646
63	-0.0730	0.7066	18.0063
64	-0.0687	0.4919	20.6707
65			

Figura 5.11 - Coeficientes A, B y C en columnas 1, 2 y 3 respectivamente

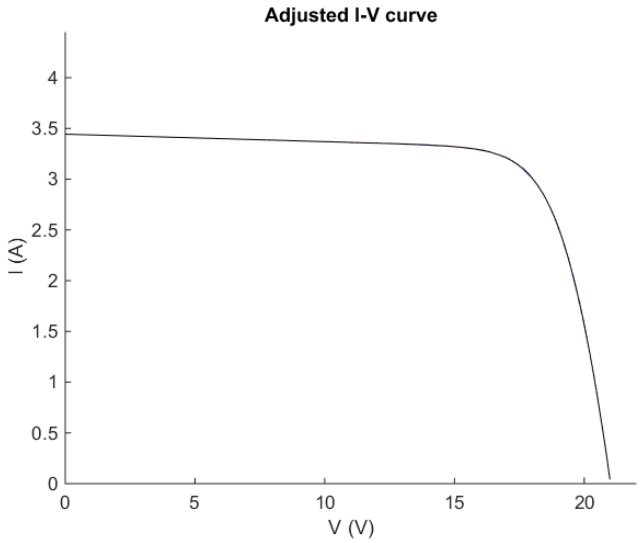


Figura 5.12 - Curva aproximación celda MSX-60

En la Figura 5.12 se puede observar la calidad de la aproximación, teniendo la curva original sobre la curva aproximada.

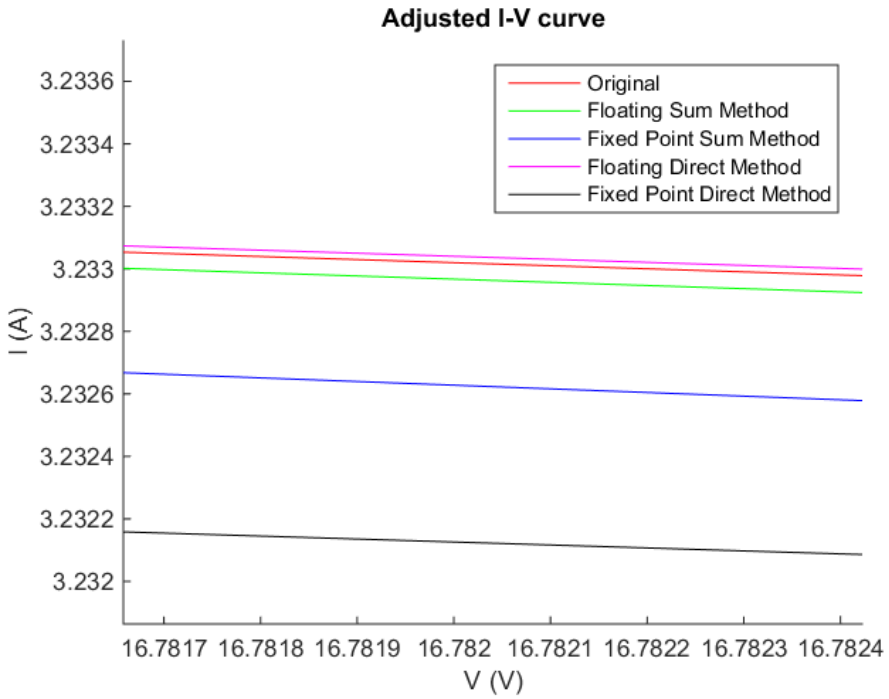


Figura 5.13 - Detalle aproximación MSX-60

En el detalle de la Figura 5.13 se puede apreciar el error existente entre la curva original validada en la sección anterior, contra la curva generada en arquitectura de punto fijo el cuál es menor al 0.01 A.

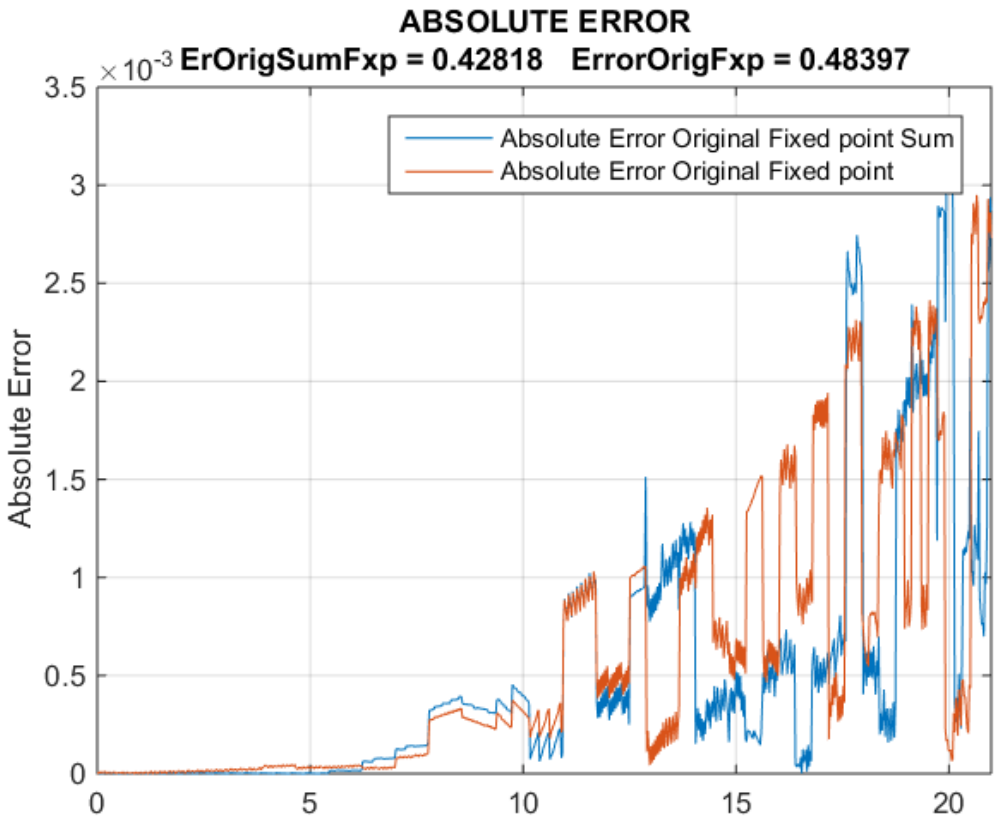


Figura 5.14 - Error absoluto MSX-60

Se obtuvo un valor de SQNR de: **71.170470**

En la Figura 5.14 se observa que el mayor valor absoluto no sobre pasa el valor de $2.9 \times 10^{-3.}$

Tomando en cuenta el valor máximo de la curva en 3.44 y el máximo error, tenemos un error máximo porcentual de: 0.084%

5.2.1.2 Celda Kyocera KG200GT

Se obtuvieron 64 valores para cada uno de los coeficientes de la ecuación de segundo grado. Se muestran en la imagen los últimos 20 valores a manera de referencia.

Editor - top_module.m

coef_ram

64x3 double

	1	2	3
41	-0.2330	12.2910	-154.7007
42	-0.2458	13.0216	-165.1919
43	-0.2556	13.5931	-173.4760
44	-0.2623	13.9850	-179.2079
45	-0.2657	14.1875	-182.1979
46	-0.2660	14.2025	-182.4193
47	-0.2633	14.0415	-179.9957
48	-0.2581	13.7239	-175.1721
49	-0.2507	13.2739	-168.2771
50	-0.2418	12.7179	-159.6832
51	-0.2316	12.0824	-149.7739
52	-0.2206	11.3921	-138.9142
53	-0.2092	10.6685	-127.4346
54	-0.1977	9.9303	-115.6200
55	-0.1862	9.1921	-103.7066
56	-0.1751	8.4657	-91.8819
57	-0.1643	7.7595	-80.2913
58	-0.1540	7.0797	-69.0408
59	-0.1443	6.4303	-58.2047
60	-0.1352	5.8137	-47.8315
61	-0.1266	5.2311	-37.9492
62	-0.1185	4.6825	-28.5693
63	-0.1111	4.1673	-19.6916
64	-0.1041	3.6847	-11.3075
65			

Figura 5.15 - Coeficientes A, B y C en columnas 1, 2 y 3 respectivamente

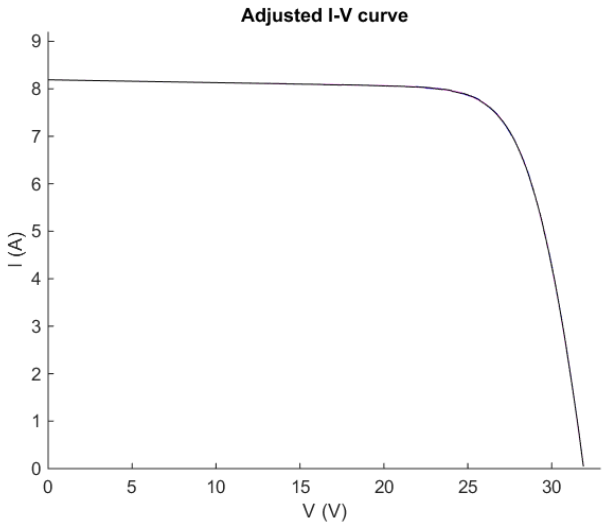


Figura 5.16 - Curva aproximación Kyocera KG200GT

En la Figura 5.16 se puede observar la calidad de la aproximación, teniendo la curva original sobre la curva aproximada.

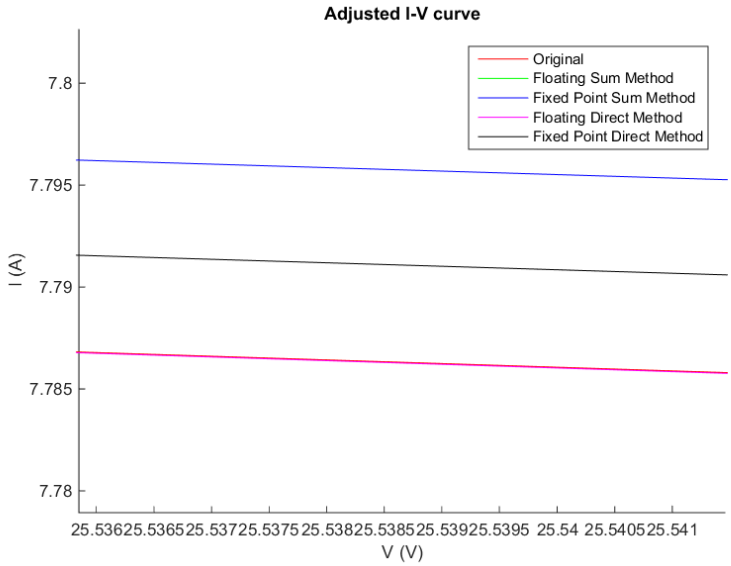


Figura 5.17 - Detalle aproximación Kyocera

Se obtuvo un valor de SQNR de: **67.088218** con un error máximo porcentual de 0.16%.

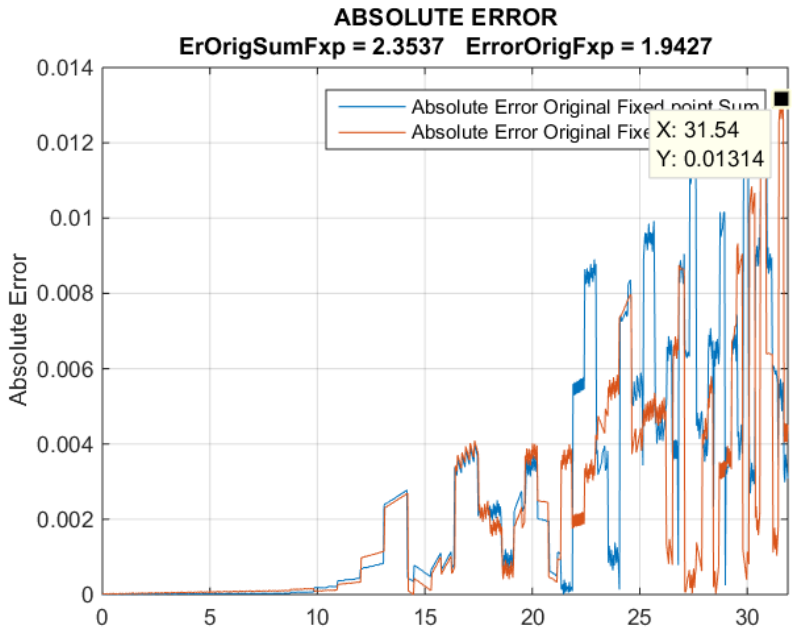


Figura 5.18 - Error absoluto Kyocera

5.2.1.3 SQ150-PC

Se muestran en la imagen los últimos 20 valores de coeficientes a manera de referencia.

coef_ram

64x3 double

	1	2	3
43	-0.0649	4.3963	-70.1824
44	-0.0660	4.4812	-71.7791
45	-0.0664	4.5101	-72.3262
46	-0.0661	4.4850	-71.8447
47	-0.0651	4.4105	-70.4038
48	-0.0636	4.2930	-68.1087
49	-0.0617	4.1396	-65.0876
50	-0.0594	3.9580	-61.4789
51	-0.0568	3.7556	-57.4206
52	-0.0542	3.5391	-53.0419
53	-0.0514	3.3144	-48.4581
54	-0.0486	3.0866	-43.7698
55	-0.0459	2.8595	-39.0583
56	-0.0432	2.6364	-34.3892
57	-0.0407	2.4195	-29.8131
58	-0.0382	2.2107	-25.3685
59	-0.0359	2.0108	-21.0803
60	-0.0337	1.8207	-16.9674
61	-0.0316	1.6405	-13.0398
62	-0.0297	1.4705	-9.3020
63	-0.0279	1.3104	-5.7544
64	-0.0262	1.1600	-2.3949
65			
66			

Figura 5.19 - Coeficientes A, B y C en columnas 1, 2 y 3 respectivamente

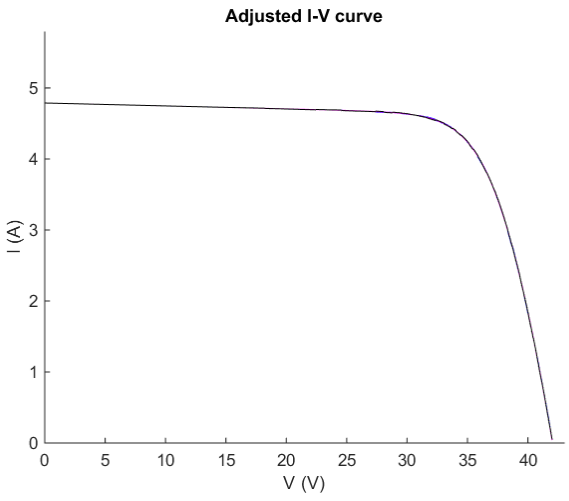


Figura 5.20 - Curva aproximación SQ150-PC

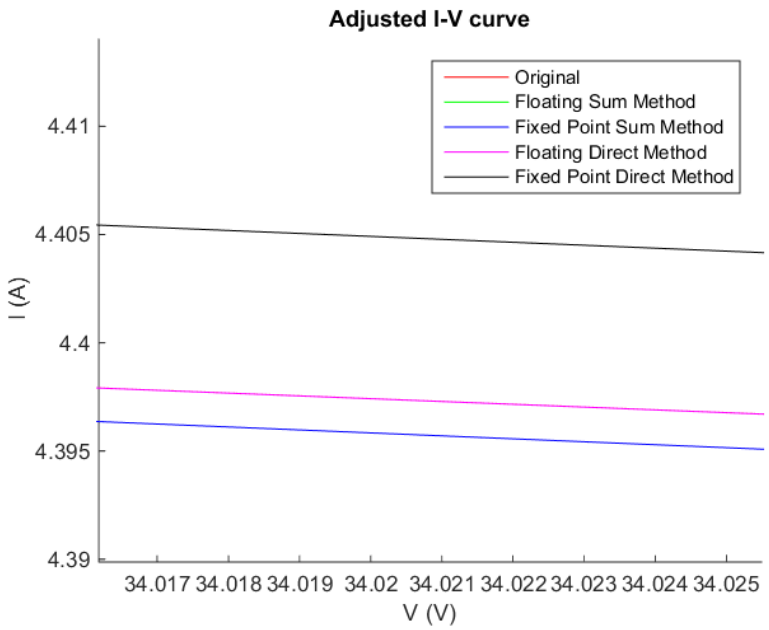


Figura 5.21 - Detalle aproximación SQ150C

Se obtuvo un valor de SQNR de: **61.544190** con un error máximo porcentual de 0.229%.

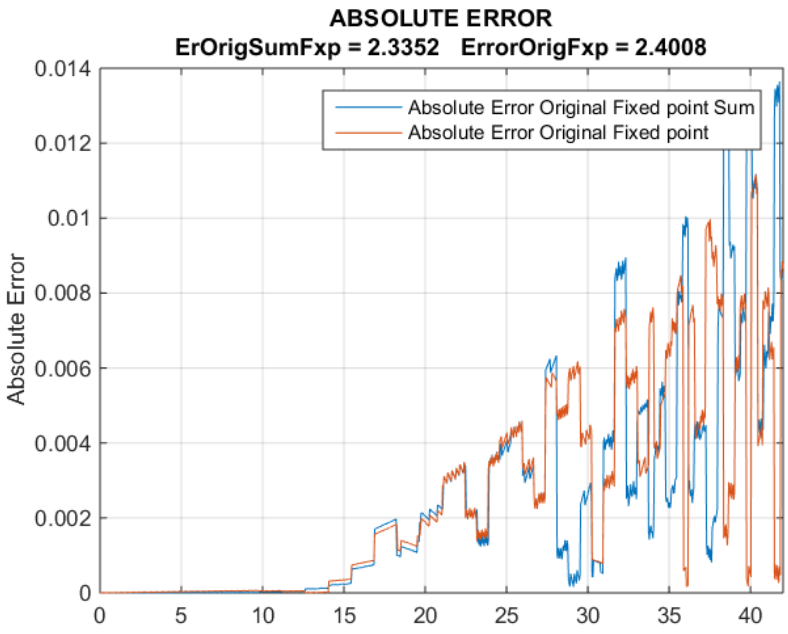


Figura 5.22 - Error abosluto SQ150C

5.2.1.4 Shell SP-70

coef_ram

64x3 double

	1	2	3
41	-0.2081	6.7205	-50.0274
42	-0.2023	6.5008	-47.9530
43	-0.1952	6.2298	-45.3714
44	-0.1871	5.9201	-42.3928
45	-0.1784	5.5833	-39.1234
46	-0.1694	5.2298	-35.6595
47	-0.1603	4.8682	-32.0850
48	-0.1512	4.5059	-28.4703
49	-0.1423	4.1484	-24.8713
50	-0.1337	3.8001	-21.3336
51	-0.1255	3.4639	-17.8892
52	-0.1177	3.1419	-14.5610
53	-0.1104	2.8355	-11.3664
54	-0.1035	2.5452	-8.3135
55	-0.0970	2.2713	-5.4084
56	-0.0910	2.0135	-2.6511
57	-0.0854	1.7714	-0.0402
58	-0.0802	1.5445	2.4278
59	-0.0753	1.3320	4.7578
60	-0.0708	1.1332	6.9561
61	-0.0667	0.9472	9.0289
62	-0.0628	0.7733	10.9831
63	-0.0592	0.6106	12.8250
64	-0.0559	0.4584	14.5624

Figura 5.23 - Coeficientes A, B y C en columnas 1, 2 y 3 respectivamente

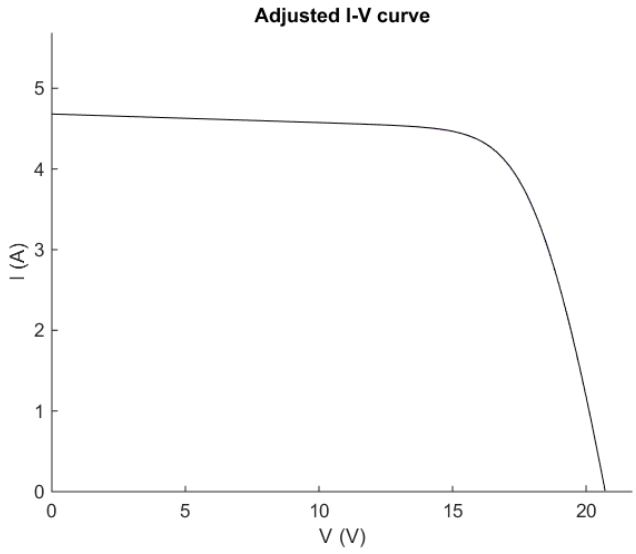


Figura 5.24 - Curva aproximación shell SP-70

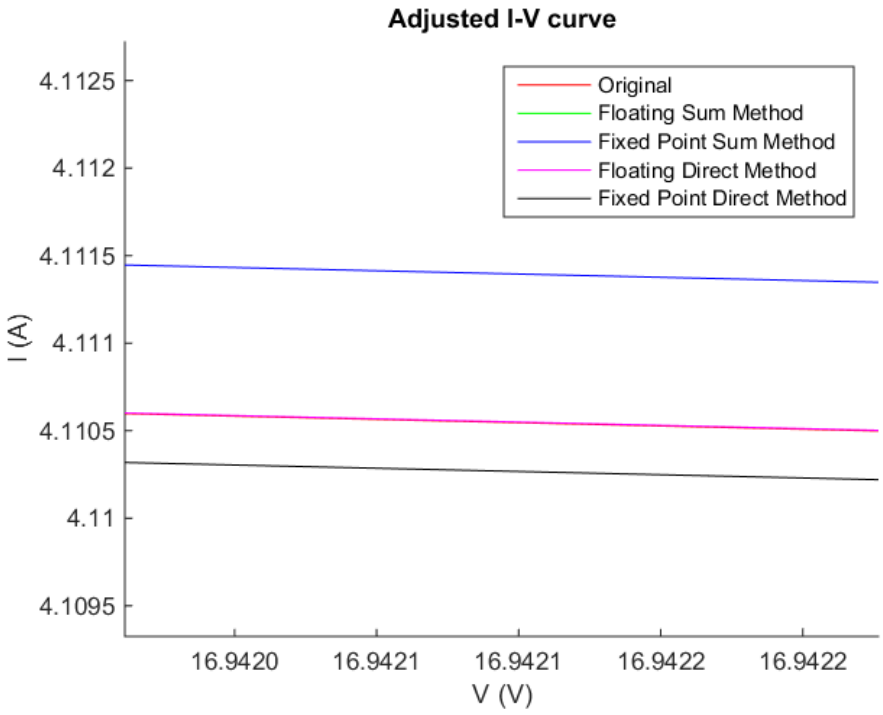


Figura 5.25 - Detalle aproximación curva Shell SP-70

Se obtuvo un valor de SQNR de: **79.149499** con un error máximo porcentual de 0.034%.

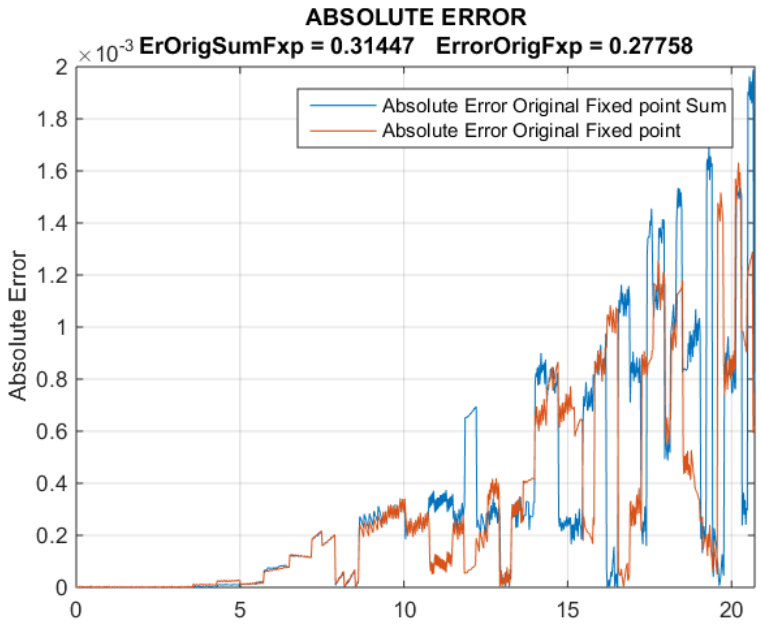


Figura 5.26 - Error absoluto Shell SP-70

5.2.1.5 Shell ST ST40

coef_ram

64x3 double

	1	2	3
42	-0.0238	0.4910	1.0377
43	-0.0222	0.4253	1.7178
44	-0.0207	0.3640	2.3585
45	-0.0194	0.3069	2.9611
46	-0.0181	0.2538	3.5273
47	-0.0170	0.2044	4.0578
48	-0.0159	0.1582	4.5583
49	-0.0149	0.1154	5.0268
50	-0.0140	0.0755	5.4669
51	-0.0132	0.0384	5.8801
52	-0.0124	0.0038	6.2690
53	-0.0117	-0.0284	6.6347
54	-0.0111	-0.0586	6.9791
55	-0.0104	-0.0867	7.3030
56	-0.0099	-0.1130	7.6087
57	-0.0094	-0.1375	7.8966
58	-0.0089	-0.1606	8.1697
59	-0.0084	-0.1822	8.4274
60	-0.0080	-0.2025	8.6714
61	-0.0076	-0.2216	8.9026
62	-0.0072	-0.2396	9.1220
63	-0.0069	-0.2565	9.3295
64	-0.0066	-0.2724	9.5271
65			

Figura 5.27 - Coeficientes A, B y C en columnas 1, 2 y 3 respectivamente

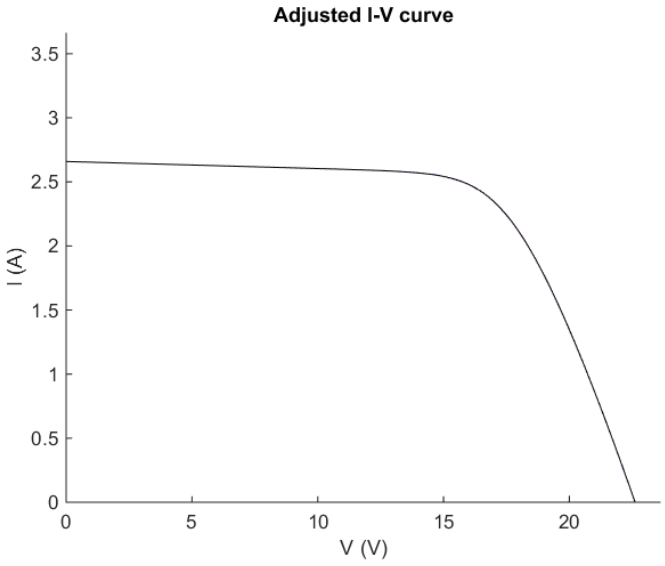


Figura 5.28 – Aproximación Shell ST40

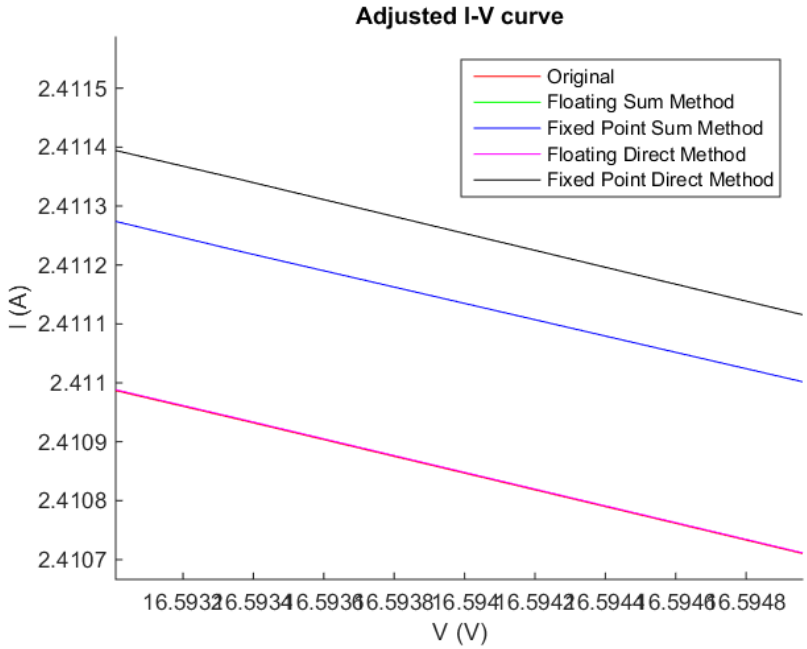


Figura 5.29 - Detalle aproximación Shell ST40

Se obtuvo un valor de SQNR de: **79.186992** con un error máximo porcentual de 0.0301%.

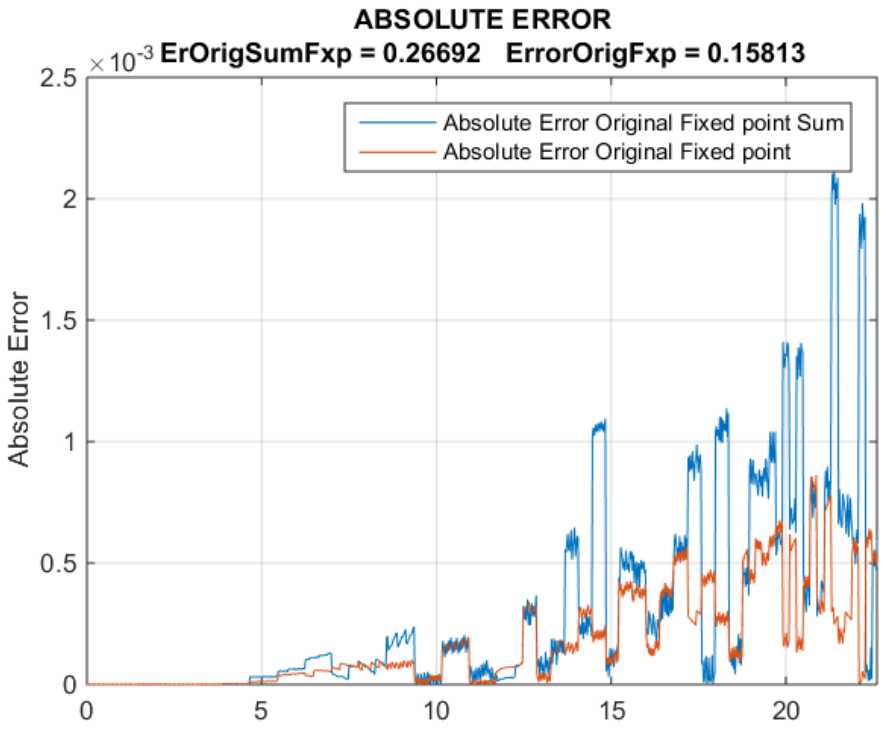


Figura 5.30 - Error absoluto Shell ST4

5.3 Evaluador de polinomios utilizando arreglos sistólicos

En la tarea de diseño de sistemas digitales siempre es importante la verificación del funcionamiento de la arquitectura a diseñar. Actualmente, existen diversas herramientas de softwares de desarrollo que permiten la verificación de la arquitectura llevando a cabo simulaciones virtuales antes de implementar el diseño en la plataforma de desarrollo.

La simulación usualmente se lleva a cabo con la misma estructura HDL. Se crea un programa especial conocido como testbench para emular el funcionamiento físico. El término testbench usualmente se refiere a un código para la simulación utilizado para crear una secuencia de entradas predeterminadas de un diseño para poder observar la respuesta de la arquitectura bajo prueba. Un testbench es comúnmente implementado utilizando los lenguajes VHDL, Verilog, System-Verilog y OpenVeram pero pueden combinarse también con archivos externos o rutinas en C.

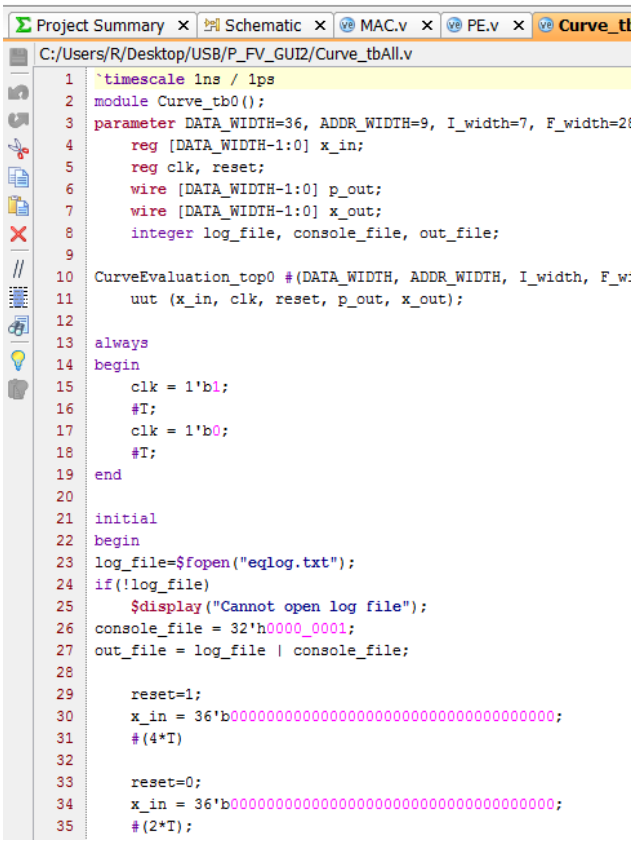


Figura 5.31 - Extracto código Testbench

Después de que el código es desarrollado, puede ser simulado en una computadora para verificar la correcta operación del circuito y pueda ser sintetizado en un dispositivo físico.

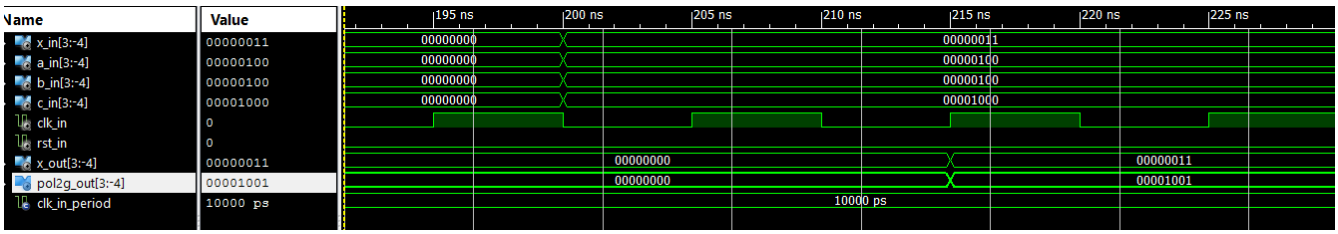


Figura 5.32 - Testbench realizado para el modulo MAC

Para probar la funcionalidad de ésta etapa se ingresaron valores controlados de coeficientes y de *x*, y se analizó la salida esperada sacando el error.

Entrada: (2.25_{10}) 0010.0100_2 (3.25_{10}) 0011.0100_2 Salida: 0111.0101_2 $[7.3125_{10}]$	Operación decimal: $2.25_{10} \times 3.25_{10} = 7.3125_{10}$ (0111.0101_2) Error: $7.3125_{10} - 7.3125_{10} = 0$
Entrada: (2.3125_{10}) 0010.0101_2 (3.1875_{10}) 0011.0011_2 Salida: 0111.0110_2 $[7.375_{10}]$	Operación decimal: $2.3125_{10} \times 3.1875_{10} = 7.37109375_{10}$ (0111.01011111_2) Redondeo: 0101 + 1 = .0110 Error: 0.00290625

Entrada: (2.1875_{10}) 0010.0011_2 (3.1875_{10}) 0011.0011_2 Salida: 0111.0000_2 $[7_{10}]$	Operación decimal: $2.1875_{10} \times 3.1875_{10} = 6.97265625_{10}$ (0110.11111001_2) Redondeo: 1111 + 1 = 1.0000 Error: 0.02734375
Entrada: (0.3125_{10}) 0000.0101_2 (0.2500_{10}) 0000.0100_2 Salida: 0000.0001_2 $[0.0625_{10}]$	Operación decimal: $0.3125_{10} \times 0.25_{10} = 0.078125_{10}$ (0110.000101_2) Redondeo: .0001 + 0 = 0.0001 Error: 0.015625

Figura 5.33 Análisis del error Testbench MAC

Con los resultados podemos observar que el error máximo obtenido para la arquitectura de punto fijo será del 50% del valor máximo del bit menos significativo.

6. Conclusiones

- La realización de este trabajo de tesis logró la implementación de un algoritmo de segmentación de doble nivel, capaz de aproximar curvas *I-V* mediante un arreglo sistólico el cuál en dos ciclos de reloj puede arrojar resultados con un error menor al 0.01%.

- La arquitectura utilizada MAC es escalable y parametrizable, es decir, puede ser ajustada en cuanto a número de bits que se utilizarán tanto en parte entera como fraccionaria, el grado de polinomio a utilizar y el número de arquitecturas en paralelo.

- Se realizó la comprobación de los resultados propuestos por la ecuación de doble-diodo demostrando su superioridad respecto a otros modelos en cuanto a precisión y exactitud con los datos otorgados por el fabricante.

- El diseño del hardware emulador fotovoltaico funciona en base a arreglos sistólicos los cuales trabajan independientemente a los demás y transmiten los resultados del procesamiento al siguiente arreglo, al tratarse un sistema lógico combinacional, se obtienen resultados en el menor tiempo posible de manera eficiente.

- Al estar utilizando un elemento sistólico como evaluador de polinomios podemos obtener el punto deseado de la curva *I-V* para cualquier valor requerido, o en caso de ser necesario realizar un barrido en un rango de voltaje para graficar segmentos de curva.

- El número total de segmentos utilizados es mucho menor contrastado con otros métodos, y representa un espacio reducido en memoria utilizado para almacenar los coeficientes de los límites de segmento.

- Dependiendo de la precisión requerida es posible reducir el número de bits utilizados gracias a la utilización de una arquitectura en punto fijo, de ésta manera haciendo posible incrementar el número de arquitecturas inmersas en un mismo chip.

- La utilización de procesos pipeline en arreglos tiene una gran ventaja sobre procesos secuenciales, ya que permite ejecutar varios procesos en diferentes arreglos para obtener resultados en cada ciclo de reloj, obteniendo al final un sistema más veloz y eficiente.

- La aproximación por polinomios utilizando segmentación de doble nivel puede ser aplicable a otras áreas en donde se utilicen ecuaciones de gran complejidad, siendo difíciles de manipular e implementar, reduciendo costos de implementación y tiempos de procesamiento, sin sacrificar precisión a la hora de entregar los resultados.

- En futuros trabajos, se pueden realizar implementaciones de dos o más celdas en un mismo FPGA logrando abarcar áreas más complejas como estudios de sombreado parcial o reconfiguración en sitio.

7. Referencias

[1] P. W. J. R. S. Roundy, «Energy Scavenging for Wireless Sensor Networks: With Special Focus On Vibrations,,» Norwell, MA, Kluwer Academic, 2004.

[2] Z. S. H. T. Kashif Ishaque n, «Simple, fast and accurate two-diode model for photovoltaic modules,» *Solar Energy Materials & Solar Cells,* nº 95, p. 586–594, 2011.

[3] J. A. a. M. Y. Jian, «Matlab/pspice hybrid simulation modeling of solar PV cell/module,» *Applied Power Electron. Conf. and Expositivion (APEC),* pp. 1244-1250, 2011.

[4] D. K. N. J. Y.T. Tan, «A model of PV generation suitable for stability analysis,» *IEEE Trans. Energy Convers,* vol. 4, nº 19, pp. 748-755, 2004.

[5] A. H. A. Kajihara, «Model of photovoltaic cell circuits under partial shading,» *Proceedings of the IEEE International Conference on Industrial Technology,* pp. 866-870, 2005.

[6] W. D. A. C. W. Xiao, «A novel modeling method for photovoltaic cells,» *Proceedings of the IEEE 35th Annual Power Electronics Specialists Conference,* pp. 1950-1956, 2004.

[7] R. N. W. S. C. Sah, «Carrier generation and recombination in p–n junctions and p–n junction characteristics,» *Proceedings of IRE,* nº 45, pp. 1228-1243, 1957.

[8] Z. S. a. S. K. Ishaque, «A comprehensive MATLAB,» *Solar Energy,* vol. 85, nº 9, p. 2217–2227, 2011.

[9] G. F. a. B. Asaei, «A New Approach for Solar Module Temperature Estimation Using the Simple Diode Model,» *IEEE TRANSACTIONS ON ENERGY CONVERSION,* vol. 26, nº 4, pp. 1118-1126, 2011.

[10] K. I. a. H. T. Zainal Salam, «An Improved Two-Diode Photovoltaic (PV),» *IEEE,* 2010.

[11] E. I. O.-R. Omar Gil-Arias, «A General Purpose Tool for Simulating the Behavior of PV Solar Cells, Modules and Arrays,» *IEEE,* vol. 08, nº 978-1-4244-2551-8, 2008.

[12] T. S. A. G. a. A. M. M. Bouzguenda*1, «Evaluating Solar Photovoltaic System Performance using MATLAB,» de *First International Conference on Renewable Energies and Vehicular Technology*, 2012.

[13] J. A. A. Q. M. O. Yuncong Jiang, «Matlab/Pspice Hybrid Simulation Modeling of Solar PV Cell/Module,» de *IEEE*, 2011.

[14] Y. G. J. G. Z. M. G. R. Md. Rabiul Islam#1, «Simulation of PV Array Characteristics and Fabrication of Microcontroller Based MPPT,» de *6th International Conference on Electrical and Computer Engineering*, Dhaka, Bangladesh, 2010.

[15] M. F. a. D. S. D.M.K. Schofield, «Low-cost solar emulator for evaluation of maximum power point tracking methods,» *ELECTRONICS LETTERS,* vol. 47, nº 3, 2011.

[16] H. C. a. K. S. P. Damla Ickilli1, «Development of a FPGA-Based Photovoltaic Panel Emulator Based on a DC/DC Converter,» de *IEEE*, 2011.

[17] c. *. H. M. b. A. M. b. H. S. c. A. Mellit a, «FPGA-based implementation of an intelligent simulator for stand-alone photovoltaic system,» de *Expert Systems with Applications*, Elsevier, 2010.

[18] M. Z. a. A. A.-H. Yousry Atia1, «Solar Cell Emulator and Solar Cell Characteristics Measurements In Dark and Illuminated Conditions,» *WSEAS Yousry Atia, TRANSACTIONS Mohamed Zahran, on SYSTEMS Abdullah Al-Hossain and CONTROL,* vol. 6, nº 4, p. 125, 2011.

[19] M. I. R. C. C. C. M. I. W. L. S. M. I. a. Dong-U Lee, «Hierarchical Segmentation for Hardware,» *IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS,* vol. 17, nº 1, pp. 103-116, 2009.